TP3 : Tableaux, listes et tris

Christophe Rose

Vendredi 15 février 2008

Dans ce TP, nous allons continuer à manipuler les tablaux et les listes, et nous verrons comment implémenter certains algorithmes de tri.

1 Tableaux

Question 1. Programmez une fonction `miroir` de type `int vect -> int vect` qui retourne un tableau. On créera un nouveau tableau.

Question 2. Programmez une fonction `decale` de type `int vect -> int vect` qui décale les éléments d'un tableau d'un cran vers la droite et qui met le dernier élément en premier. On utilisera le même tableau.

Question 3. Programmez une fonction `pascal` de type `int -> int vect vect` telle que `pascal n` construit un triangle de Pascal à $n+1$ lignes. On obtient alors quelque chose de la forme :

```
#pascal 3;;
- : int vect vect =
 [|[|1|]; [|1; 1|]; [|1; 2; 1|]; [|1; 3; 3; 1|]|]
```

2 Types objets

On peut définir de nouveaux types en Caml grâce aux constructions objet. Ainsi, la commande `type complexe = {Re: float ; Im: float}` permet de créer le type « nombre complexe ».

On peut définir des objets de ce type par `let I = {Re: 0. ; Im: 1.}`. On récupère les parties réelle et imaginaire d'un nombre complexe `z` en utilisant `z.Re` et `z.Im`.

Question 4. Programmez l'addition, la multiplication, la soustraction, le conjugué, le module, l'inverse et la division pour le type « nombre complexe » ainsi créé.

Question 5. Créez un type pour $\mathbb{Z}[\sqrt{5}]$, et programmez la multiplication, ainsi que la puissance. Avec la relation $\sqrt{5} \cdot F_n = \left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n$, déduisez-en une fonction qui calcule rapidement la suite de Fibonacci. Le nombre total de multiplications effectuées ne devra pas dépasser $\log_2 n$ environ.

3 Listes

Question 6. Écrivez une fonction `make_list` de type `int -> 'a -> 'a list` telle que `make_list n x` donne une liste contenant n fois la valeur x.

Question 7. Écrivez une fonction `longueur_liste` qui détermine la taille d'une liste.

Question 8. Est-ce que toutes les listes ont une taille bien définie ?

Question 9. Écrivez une fonction `appartient` de type `'a -> 'a list -> bool` qui détermine si un élément appartient à une liste.

Question 10. Écrivez une fonction `aplatit` de type `'a list list -> 'a list` qui prend une liste de listes et qui donne le concaténé de ces listes. On commencera par écrire une fonction auxiliare `concatene`.

Question 11. Écrivez une fonction récursive `pgcd` de type `int * int -> int` qui calcule le PGCD de deux entiers. On utilisera l'algorithme d'Euclide et l'opérateur `mod`.

Question 12. Programmez une fonction itérative `premier` de type `int -> bool` qui détermine si un entier n est premier ou non. (On pourra faire en sorte de n'utiliser qu'au plus $\sqrt{n}$ fois l'opérateur `mod`.)

Question 13. À l'aide de la fonction `premier` définie ci-dessus, écrivez une fonction `decompose` de type `int -> int list` qui décompose un entier en produit de nombres premiers. Par exemple,

```
# decompose 36;;
- : int list = [2; 2; 3; 3]
```

4 Algorithmes de tri

Question 14. Écrivez une fonction `random_tab` de type `int -> int vect` qui crée un tableau de taille donnée rempli d'entiers aléatoires. On utilisera la commande `random__int 1000000000;;`.

Question 15. Écrivez une fonction `est_trie` de type `'a vect -> bool` qui détermine si un tableau est trié dans l'ordre croissant.

Question 16. Écrivez une fonction `algo_ok` de type `('a vect -> 'a vect) -> int -> bool` qui vérifie si une fonction sensée trier un tableau fonctionne sur un tableau aléatoire de taille donnée.

Il existe trois types de tris « en $O(n^2)$ ». Ils s'agit du tri insertion, du tri sélection et du tri à bulles.

Pour effectuer le tri insertion, on considère successivement tous les éléments d'un tableau. On les insère un par un dans un autre tableau, de sorte que ce dernier tableau soit toujours trié. À la fin, le deuxième tableau est trié et contient les mêmes éléments que le premier.

3	*1*	*5*	*7*	*4*	*8*
1	*3*	*5*	*7*	*4*	*8*
1	**3**	*5*	*7*	*4*	*8*
1	**3**	**5**	*7*	*4*	*8*
1	**3**	**4**	**5**	*7*	*8*
1	**3**	**4**	**5**	**7**	*8*
1	**3**	**4**	**5**	**7**	**8**

Pour effectuer le tri sélection, on cherche le plus petit élément du tableau, qu'on déplace en première position. Puis on cherche le deuxième plus petit, qu'on déplace en deuxième position. À la fin, tous les éléments sont à leur place.

3	*1*	*5*	*7*	*4*	*8*
1	*3*	*5*	*7*	*4*	*8*
1	**3**	*5*	*7*	*4*	*8*
1	**3**	**4**	*7*	*5*	*8*
1	**3**	**4**	**5**	*7*	*8*
1	**3**	**4**	**5**	**7**	*8*
1	**3**	**4**	**5**	**7**	**8**

Pour effectuer le tri à bulles, on considère tous les couples d'éléments successifs, en partant du début du tableau. Si un couple n'est pas trié, on les échange. Lorsqu'on a parcouru tout le tableau, on sait que le plus grand élément est à la fin. À la fin, tous les éléments ont remonté à leur position.

3	*1*	*5*	*7*	*4*	*8*
1	*3*	*5*	*7*	*4*	*8*
1	*3*	*5*	*4*	*7*	*8*
1	*3*	*5*	*4*	**7**	**8**
1	*3*	*4*	*5*	**7**	**8**
1	*3*	*4*	**5**	**7**	**8**
1	**3**	**4**	**5**	**7**	**8**

Question 17. Écrivez une fonction `echange` telle que `echange v a b` échange les valeurs `v.(a)` et `v.(b)`.

On utilisera le même tableau pour le départ et l'arrivée.

Question 18. Programmez une fonction `trouve_pos` telle que `trouve_pos v x k` trouve l'entier j compris entre 0 et $k-1$ tel que `x` soit compris entre `v.(j)` et `v.(j+1)`, en supposant que le tableau `v` soit trié entre les indices 0 et k. En déduire une fonction `tri_insert` de type `'a vect -> 'a vect` qui effectue le tri insertion.

Question 19. Programmez une fonction `plus_petit` telle que `plus_petit v k` trouve l'indice du plus petit élément de `v` d'indice supérieur ou égal à k. En déduire une fonction `tri_select` de type `'a vect -> 'a vect` qui effectue le tri sélection.

Question 20. Programmez une fonction `tri_bulle` de type `'a vect -> 'a vect` qui effectue le tri à bulles.

Question 21. Vérifiez que les fonctions `tri_insert`, `tri_select` et `tri_bulle` fonctionnent convenablement sur certains tableaux, de tailles variant entre 0 et 4000000.

Question 22. Refaites toutes les questions en remplaçant les tableaux par des listes.

5 Matrices

Une matrice de type `'a` est un tableau de tableaux de type `'a`.

On ne peut pas créer une matrice simplement par les commandes suivantes :

```
# let A =
  let v = make_vect nb_colonnes 0 in
  make_vect nb_lignes v ;;
```

Sinon, la commande `A.(0).(0)<-1` serait l'équivalent de `v.(0)<-1` et modifierait toute la première ligne et pas seulement le premier élément de la matrice.

Il faut donc recréer avec une boucle tous les vecteurs ligne.

```
# for i = 0 to (nb_lignes - 1) do
  let v = make_vect nb_colonnes 0 in
  A.(i)<-v;
  done;
  ;;
```

Question 23. Écrivez une fonction `dim` de type `'a vect vect -> int * int`, qui prend un matrice en argument et qui renvoie son nombre de lignes et son nombre de colonnes. Si l'argument n'est pas une matrice (pensez au triangle de Pascal), la fonction renvoie `(-1, -1)`.

Question 24. Écrivez des fonctions pour les opérations suivantes sur les matrices : addition, multiplication, transposée, trace, déterminant, inverse.