TP4 : Récursivité sur les listes et tris

Christophe Rose

Vendredi 14 mars 2008

Dans ce TP, nous allons écrire des programmes récursifs sur des listes, et nous verrons comment implémenter certains algorithmes de tri.

1 Listes

Pour montrer un théorème par récurrence sur tous les entiers $n \in \mathbb{N}$, il suffit de montrer deux choses : que ce théorème est vrai pour $n = 0$, et que s'il est vrai pour n quelconque, il est vrai pour $n + 1$.

Lorsqu'on écrit un algorithme récursif, on souhaite savoir si l'argument est terminal ou non, tout en retirant des informations à cet argument. On utilise alors le *pattern matching* ou recherche de motifs.

```
let rec fibo_naif n = match n with
  0 -> 0
| 1 -> 1
| _ -> fibo_naif (n-1) + fibo_naif (n-2) ;;
```

Question 1. Écrivez une fonction récursive `make_list` de type `int -> 'a -> 'a list` telle que `make_list n x` donne une liste contenant n fois la valeur x.

Pour trouver la tête et la queue d'une liste non vide `l`, on peut utiliser `hd l` et `tl l`. On peut également utiliser `let t::q=l in (t,q)`. Pour une liste quelconque, le *pattern matching* permet d'écrire les fonctions de manière élégante.

```
let rec somme l = match l with
| [] -> 0
| t::q -> t+(somme q);;
```

Question 2. Écrivez une fonction récursive `longueur_liste` qui détermine la taille d'une liste.

Question 3. Est-ce que toutes les listes ont une taille bien définie ?

Question 4. Écrivez une fonction récursive `appartient` de type `'a -> 'a list -> bool` qui détermine si un élément appartient à une liste.

Question 5. Écrivez une fonction récursive `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 6. Écrivez une fonction récursive `minmax` de type `'a list -> 'a * 'a` qui renvoie le minimum et le maximum d'une liste non vide.

Question 7. Écrivez une fonction récursive `occmin` de type `'a list -> 'a * int` qui renvoie le minimum d'une liste non vide et le nombre d'occurences de ce minimum dans la liste.

2 Récursivité

Question 8. 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 9. 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 10. 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]
```

3 Algorithmes de tri

Question 11. 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 12. Écrivez une fonction `est_trie` de type `'a vect -> bool` qui détermine si un tableau est trié dans l'ordre croissant.

Question 13. 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 14. 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 15. 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 16. 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 17. Programmez une fonction `tri_bulle` de type `'a vect -> 'a vect` qui effectue le tri à bulles.

Question 18. Vérifiez que les fonctions `tri_insert`, `tri_select` et `tri_bulle` fonctionnent convenablement sur certains tableaux, de tailles variant entre 0 et 4000000.

Question 19. Refaites toutes les questions en remplaçant les tableaux par des listes.

4 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 20. 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 21. Écrivez des fonctions pour les opérations suivantes sur les matrices : addition, multiplication, transposée, trace, déterminant, inverse.