

# Révisions et automatismes

OPTION INFORMATIQUE - TP n° 3.0 - Olivier Reynet

## À la fin de ce chapitre, je sais :

- ☞ implémenter les opérations élémentaires récursives sur les listes
- ☞ utiliser les vecteurs et la programmation impérative
- ☞ implémenter le tri par insertion, le tri fusion et le tri rapide
- ☞ implémenter les opérations élémentaires récursives sur les arbres binaires
- ☞ effectuer une démonstration par récurrence
- ☞ effectuer une démonstration par induction structurale

## A Listes et récursivité

Écrire des fonctions OCaml qui implémentent les opérations suivantes sur les listes :

- A1. Calculer la longueur d'une liste (`length`).
- A2. Déterminer le *i*-ème élément d'une liste (`at` ou `nth`). On considèrera deux solutions : la première échoue en lançant une exception si l'élément n'est pas trouvé. La seconde renvoie un type `option`.
- A3. Tester l'appartenance d'un élément à une liste (`mem`).
- A4. Concaténer de deux listes (`concat`).
- A5. Aplatir une liste de listes (`flatten`).
- A6. Appliquer une fonction à tous les éléments d'une liste et retourner la liste correspondante : essayer de le faire en temps linéaire (`map`).
- A7. Extraire les deux derniers éléments d'une liste. Cette fonction renvoie une tuple. Fonctionne-t-elle sur une liste de liste ?
- A8. Construire la liste inverse d'une liste, c'est à dire que l'élément 0 devient le dernier élément, le premier l'avant dernier...

## B Vecteurs et programmation impérative

- B1. Écrire une fonction qui calcule le *n*-ième terme de la suite de Fibonacci en utilisant un vecteur où sont stockées les valeurs successives. Quelle est sa complexité temporelle? spatiale?
- B2. Écrire une fonction qui calcule le *n*-ième terme de la suite de Fibonacci en utilisant des références. Quelle est sa complexité temporelle? spatiale?

- B3. Écrire une fonction `mem x tab` qui test l'appartenance d'un élément `x` à un vecteur `tab`. Quelle est sa complexité? Peut-on faire mieux si l'on suppose le tableau trié? Programmer une version efficace dans ce cas.
- B4. Écrire une fonction `v_concat v1 v2` qui concatène deux tableaux unidimensionnels et renvoie le nouveau tableau ainsi créé.
- B5. Écrire une fonction `m_concat m1 m2` qui concatène deux matrices ayant le même nombre de lignes.
- B6. Écrire une fonction `array_map f tab` qui, sur le modèle de la fonction `map` pour les listes, renvoie un nouveau vecteur contenant les images des éléments de `tab` par la fonction `f`.

## C Tris

- C1. Écrire une fonction qui effectue le tri par insertion. On décompose le problème comme suit :
- (a) Écrire une fonction récursive de prototype `val insert_elem : 'a list -> 'a -> 'a list` qui insère un élément dans une liste triée à la bonne place.
  - (b) Écrire une fonction de prototype `val insert_sort : 'a list -> 'a list` qui trie par insertion une liste.
  - (c) Utiliser les fonctions `List.fold_left` et `insert_elem` pour écrire une fonction le tri par insertion en une seule ligne.
- C2. Discuter la terminaison, la correction et la complexité de la fonction `insert_elem`.
- C3. Discuter la terminaison, la correction et la complexité de la fonction `insert_sort`.
- C4. Écrire une fonction qui réalise le tri fusion. On décompose le problème comme suit :
- (a) une fonction qui sépare une liste en deux,
  - (b) une fonction qui réalise la fusion de deux listes triées en une seule liste triée,
  - (c) une fonction qui effectue le tri fusion.
- C5. Discuter la terminaison, la correction et la complexité du tri fusion.
- C6. Écrire une fonction qui réalise le tri rapide. On décompose le problème comme suit :
- (a) une fonction qui partitionne une liste en deux d'après le pivot,
  - (b) une fonction qui effectue le tri rapide.
- C7. Discuter la terminaison, la correction et la complexité du tri rapide.

## D Arbres binaires de recherche

On considère la définition récursive du type `'a arbre` :

```
1  type 'a tree = Nil | Node of 'a tree * 'a * 'a tree ;;
```

---

Écrire les fonctions qui implémentent les opérations suivantes sur un type `tree` :

- D1. Calculer la hauteur d'un arbre binaire.
- D2. Calculer le nombre total de nœuds d'un arbre binaire.
- D3. Calculer le nombre de feuilles d'un arbre binaire.
- D4. Calculer récursivement le nombre de nœuds internes d'un arbre binaire. Démontrer par induction structurelle que cette fonction est correcte.
- D5. Ajouter un élément à la fin de la branche la plus à gauche et renvoyer le nouvel arbre.