

# Option Informatique

## TP 3: Programmation dynamique

David Baelde

david.baelde@ens-lyon.org

La programmation dynamique est un principe général d'optimisation algorithmique. Il peut être utile quand on calcule une quantité définie récursivement. Il s'agit essentiellement de mémoriser les valeurs de retour des appels à la fonction calculant notre quantité, pour éviter de recalculer plusieurs fois la même chose. C'est donc analogue à l'option `remember` de Maple.

Les exercices suivants tirent parti de l'utilisation de tableaux, un des aspects *impératifs* d'OCaml.

### 1 Préliminaires

► Évaluez la complexité des deux fonctions classiques suivantes.

```
let rec f n = if n < 2 then 1 else n * (f (n-1))
let rec f n = if n < 2 then 1 else (f (n-1))+(f (n-2))
```

Supposons qu'on ait à calculer  $v(n)$  en fonction de certaines valeurs  $v(i)$  pour  $i$  inférieur à  $n$ . L'approche par programmation dynamique consiste à écrire l'algorithme selon le schéma suivant :

```
let f n =
  let memo = Array.create (n+1) None in
  let rec aux n =
    match memo.(n) with
    | None ->
      let vn = (* calcule v(n) par appels récursifs à aux *) in
      memo.(n) <- vn ;
      vn
    | Some vn -> vn
  in
  aux n
```

- ▶ Grâce à cette technique quelle est maintenant la complexité qu'on peut obtenir pour le calcul de la factorielle et de la suite de Fibonacci ? Programmez la version ainsi optimisée pour Fibonacci.
- ▶ La programmation dynamique est une recette générale donc facilement améliorable. Améliorez votre fonction de calcul de Fibonacci pour diminuer l'occupation mémoire (*exit* le tableau de taille  $n$ ) sans perdre en complexité temporelle. Peut-être même saurez vous trouver une solution encore plus rapide utilisant le calcul sur les `float` ?

## 2 Plus longue sous-liste commune

Nous traitons ici un problème classique pour lequel la programmation dynamique va enfin vraiment nous servir. Il s'agit de trouver la taille de la plus longue sous-liste commune à deux listes. On appelle sous-liste de  $[x_1; \dots; x_n]$  une liste  $[x_{p_1}; \dots; x_{p_m}]$  tel que :

$$\forall i \in \llbracket 1, m \rrbracket, \quad p_i \in \llbracket 1, n \rrbracket$$

$$\forall i \in \llbracket 1, m \rrbracket, \forall j \in \llbracket i + 1, m \rrbracket, \quad p_i < p_j$$

En français : on enlève certains éléments de la liste originale, sans changer l'ordre.

- ▶ Programmez une fonction prenant deux listes et indiquant si la première est une sous-liste de la seconde.

### 2.1 Méthode naïve

- ▶ Trouvez une équation de récurrence décrivant l'ensemble des sous-listes d'une liste donnée. Ecrivez une fonction renvoyant la liste des sous-listes d'une liste.

Dans les deux questions suivantes vous utiliserez des fonctions de la librairie standard pour aller plus vite : cherchez pour cela la documentation des fonctions `List.mem`, `List.map`, `List.foldLeft` et `List.filter`. Vous pouvez utiliser la doc en ligne ou man `List`.

- ▶ Programmez une fonction renvoyant le *max* d'une liste d'entiers positifs, et 0 si elle est vide. Vous pouvez le faire en une ligne si vous utilisez les bonnes fonctions de la librairie standard.
- ▶ Programmez de même une fonction naïve calculant la longueur de la plus longue sous-liste commune à deux listes données. Quelle est sa complexité ?

### 2.2 Méthode plus efficace

Étant données deux listes  $m = [x_1; \dots; x_p]$  et  $n = [y_1; \dots; y_q]$ , on note  $l(i, j)$  la longueur de la plus longue sous-liste commune à  $[x_1; \dots; x_i]$  et  $[y_1; \dots; y_j]$ .

- Montrez la relation suivante :

$$l(i, j) = \max \left( \begin{array}{l} l(i-1, j-1) + \delta(x_i = y_j), \\ l(i-1, j), \\ l(i, j-1) \end{array} \right)$$

On peut généraliser le schéma de programmation dynamique proposé plus haut, en remplaçant le tableau stockant les résultats par une matrice. Cependant, il faut prendre garde à certains pièges avec les matrices en Caml.

- Quelle est la différence entre `Array.create n (Array.create m None)` et `Array.make_matrix n m None` ? Essayez de modifier une cellule du premier tableau, et observez le résultat.
- Écrivez un algorithme calculant la longueur de la plus longue sous-liste commune par programmation dynamique. Quelle est sa complexité ?
- Étendez votre programme pour renvoyer une sous-liste commune de longueur maximale, ou toutes les sous-listes communes maximales.