# Programmation dynamique : reconstruction d'une solution optimale

Florian Bourse

Ce sujet propose 2 problèmes à traiter au choix en OCaml, en C, ou les deux.

## 1 Chemin de poids minimal

Étant donné une matrice de taille $n \times m$ contenant des entiers, on cherche le chemin qui relie le coin supérieur gauche au coin inférieur droit en ne se déplaçant que vers le bas ou vers la droite, et dont la somme est minimale. Dans l'exemple suivant, la somme du chemin optimal est 2427.

$$\begin{pmatrix} 131 & 673 & 234 & 103 & 18 \\ 201 & 96 & 342 & 965 & 150 \\ 630 & 803 & 746 & 422 & 111 \\ 537 & 699 & 497 & 121 & 956 \\ 805 & 732 & 524 & 37 & 331 \end{pmatrix}$$

```
let exemple = [|
[| 131; 673; 234; 103; 18 |];
[| 201; 96; 342; 965; 150 |];
[| 630; 803; 746; 422; 111 |];
[| 537; 699; 497; 121; 956 |];
[| 805; 732; 524; 37; 331 |];
|];;
```

```
int exemple[25] = {
        131, 673, 234, 103, 18,
        201, 96, 342, 965, 150,
        630, 803, 746, 422, 111,
        537, 699, 497, 121, 956,
        805, 732, 524, 37, 331,
};
```

https://projecteuler.net/problem=81

**Question 1.** Que peut-on dire des sous-chemins d'un chemin optimal allant de $(0, 0)$ à $(i, j)$ ?

Étant donné une matrice $M = (m_{i,j}) \in \mathbb{R}^{n \times m}$ (supposons que nous utilisons les indices de 0 à $n-1$ et de 0 à $m-1$), notons $c(i, j)$ la somme minimale d'un chemin allant de la case $(0, 0)$ à la case $(i, j)$.

**Question 2.** Quelle est la valeur de $c(0, 0)$ ? $c(i, 0)$ ? $c(0, j)$ ?

**Question 3.** Donner une relation de récurrence liant $c(i, j)$ à $c(i-1, j)$ et $c(i, j-1)$.

**Question 4.** Écrire une fonction de type `int array array -> int array array` ou équivalent en C qui prend en entrée une matrice $M$, et qui renvoie la matrice $C$ contenant dans la case $(i, j)$ la somme minimale d'un chemin allant de la case $(0, 0)$ à la case $(i, j)$ dans la matrice $M$.
*On utilisera une méthode itérative de programmation dynamique, de bas en haut.*

**Question 5.** Cette fonction nous permet de déterminer la somme d'un chemin optimal, mais ne nous donne pas les différentes étapes du chemin. Nous devons la modifier pour stocker plus d'informations, par exemple le chemin utilisé pour arriver à la case $(i, j)$. Cette solution est inefficace dans son utilisation de la mémoire car on stocke une information redondante. On peut reconstruire le chemin entier si on connaît pour chaque case $(i, j)$ la case précédente dans un chemin optimal.

Modifier votre fonction pour renvoyer également une autre matrice donnant pour chaque case $(i, j)$, la case précédente dans un chemin de somme minimale entre $(0, 0)$ et $(i, j)$.

**Question 6.** Écrire une fonction récursive qui prend en entrée la matrice définie précédemment ainsi qu'un couple $(i, j)$ et qui affiche la liste des cases à suivre pour aller de $(0, 0)$ à $(i, j)$ en suivant un chemin de somme minimale.

*Bonus : écrire une fonction qui calcule le nombre de chemins de* $(0, 0)$ *à* $(n-1, m-1)$ *pour comparer la complexité de votre algorithme à une recherche naïve.*

# 2 Ordonnancement d'intervalles pondérés

Étant donné un ensemble d'intervalles $I_i = [a_i; b_i[$, dont chacun possède une valeur $v_i$. On veut choisir un sous-ensemble d'intervalles mutuellement compatibles de valeur maximum. Deux intervalles sont compatibles si leur intersection est vide.

— En OCaml : On représentera une instance de ce problème par une liste de triplets $(a_i, b_i, v_i)$ de type `(float * float * float) list`.

— En C : On représentera une instance de ce problème par trois tableaux de `double` $\mathbf{a}, \mathbf{b}, \mathbf{v}$ ainsi que leur taille $n$.

Le jeu d'exemple suivant a pour poids maximum 120., en utilisant le deuxième, le troisième et le cinquième intervalle.

```
let exemple = [
  (0.,10.,100.);
  (1.,2.,40.);
  (2.,3.,40.);
  (2.5,4.5,50.);
  (3.2,5.,40.);
];;
```

C

```
double a[5] = {0.,1.,2.,2.5,3.2};
double b[5] = {1.,2.,3.,4.5,5.};
double v[5] = {100.,40.,40.,50.,40.};
```

**Question 7.** On va commencer par trier les intervalles par date de fin décroissante en OCaml, ou croissante en C.

— En OCaml : Écrire une fonction qui prend en entrée une liste de triplets $(a_i, b_i, v_i)$ et qui renvoie le triplet $(a_i, b_i, v_i)$ qui possède la plus grande valeur de $b_i$, ainsi que le reste de la liste. Puis l'utiliser pour écrire une fonction qui prend en entrée une liste de triplets et qui renvoie une liste contenant les mêmes triplets, mais trié par ordre de $b_i$ décroissant.

— En C : Écrire une fonction qui parcours une liste et renvoie l'indice du maximum. Écrire une fonction qui prend en argument les trois tableaux $\mathbf{a}, \mathbf{b}, \mathbf{v}$ ainsi que deux indices $i$ et $j$, et qui échange les valeurs des nombres en position $i$ et en position $j$ dans les trois tableaux.Les utiliser pour trier la liste $\mathbf{b}$, en changeant de place simultanément les valeurs des tableaux $\mathbf{a}$ et $\mathbf{v}$.

**Question 8.** On va maintenant établir la relation de récurrence qui va nous permettre de résoudre le problème. Étant donné un intervalle $(a_i, b_i, v_i)$ de la liste, on peut soit l'ajouter à notre ensemble d'intervalles, soit passer au suivant. Si on choisit de garder cet intervalle, les intervalles qui sont compatibles avec celui là sont les $(a_j, b_j, v_j)$ dont $b_j < a_i$. Comme on a classé les intervalles avant, on peut filtrer efficacement les intervalles qui nous intéressent.

- En OCaml : Écrire une fonction qui prend en entrée une liste d'intervalles triée par valeur de $b_j$ décroissante et une valeur $a_i$ et qui renvoie la liste des intervalles qui ont une valeur $b_j < a_i$.
- En C : Écrire une fonction qui prend en entrée les trois tableaux $\mathbf{a}, \mathbf{b}, \mathbf{v}$ tels que $\mathbf{b}$ est trié ainsi que leur taille $n$ et un indice $i$ et qui renvoie le plus grand indice $j$ tel que $b_j < a_i$, ou -1 si il n'existe pas.

**Question 9.** Écrire une fonction qui permet de calculer la valeur maximal atteignable pour un ensemble d'intervalles donnés.

**Question 10.** Adapter la fonction pour afficher ou renvoyer les intervalles qui satisfont cette valeur.