TD 9 - Programmation dynamique.

Exercice 1.

On reprend l'exercice sur le rendu de monnaie du TD précédent. On rappelle qu'étant donnés une somme à rendre n et un ensemble de pièces dont les valeurs sont choisies dans $\{p_1, p_2, \ldots, p_k\}$, on veut rendre la monnaie avec le moins de pièces possibles. On suppose de plus que cet ensemble contient 1, de sorte qu'il est toujours possible de rendre la somme n, et qu'on a toujours suffisamment de pièces de chaque valeur pour rendre la monnaie quelle que soit la solution proposée. On veut maintenant trouver un algorithme de programmation dynamique donnant le nombre minimum de pièces à utiliser pour représenter n dans n'importe quel système monétaire (même le britannique qui posait problème dans l'exercice du TD précédent).

1. Montrez la propriété de sous-structure optimale de la solution. Donnez l'équation de programmation dynamique correspondante.
2. Écrivez un algorithme de programmation dynamique prenant en entrée une somme à rendre n et un système monétaire S résolvant ce problème. Quelle est la complexité de l'algorithme ?

Exercice 2.

Soient $u = u_1 \cdots u_n$ et $v = v_1 \cdots v_m$ deux mots de taille n et m. On veut déterminer leur plus long facteur (ou bloc) commun (avec des lettres contigües), c'est-à-dire que l'on veut trouver des indices i, j et k tels que $u_i \cdots u_{i+k} = v_j \cdots v_{j+k}$ avec k maximal.

1. Soient $u =$ *anticonstitutionnellement* et $v =$ *consternations*. Quel est leur plus long facteur commun ?
2. De quelle sous-structure optimale a-t-on besoin pour résoudre ce problème ? Caractérisez-la par une équation. Déduisez la longueur maximale d'un facteur commun en fonction de cette sous-structure optimale.
3. Écrivez un algorithme de programmation dynamique prenant en entrée deux mots u et v et renvoyant la longueur maximale d'un facteur commun. Quelle est sa complexité ?
4. Modifiez votre algorithme pour qu'il renvoie également les indices i, j et k. La complexité est-elle modifiée ?

Exercice 3.

1. Écrivez un algorithme de programmation dynamique prenant en entrée un tableau et renvoyant les indices de début et de fin de sa plus longue suite croissante consécutive — c'est-à-dire que les indices de cette suite doivent être consécutifs. Par exemple, l'algorithme devra renvoyer (6, 9) pour la séquence $S = (3, 6, 5, 1, 9, 3, 2, 3, 4, 5, 1)$, ce qui correspond à la sous-séquence (2, 3, 4, 5) de longueur maximale 4.
2. Écrivez un algorithme de programmation dynamique prenant en entrée un tableau et renvoyant sa plus longue sous-suite strictement croissante. Ici, on n'exige donc plus que les indices de la solution soient consécutifs. Par exemple, si la séquence est $S = (3, 6, 5, \mathbf{1}, 9, 3, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, 1)$, l'algorithme renverra la sous-suite formée par les termes en gras, de longueur maximale 5.

Exercice 4.

On dispose d'un tableau de n entiers. On veut déterminer dans cette suite d'entiers un bloc (ou facteur) non vide d'entiers consécutifs dont la somme est maximale.

1. Quel est le bloc non vide de somme maximale dans le tableau $[5, 15, -30, 10, -5, 40, 10]$?
2. De quelle sous-structure optimale avez-vous besoin pour résoudre le problème plus efficacement qu'avec l'algorithme naïf ? Donnez une caractérisation (avec une équation) de cette sous-structure optimale. Déduisez-en la somme maximale cherchée en fonction de cette sous-structure optimale.
3. Écrivez un algorithme de programmation dynamique prenant en entrée un tableau et renvoyant la somme maximale de ses blocs non vides. Quelle est sa complexité ?
4. Modifiez l'algorithme précédent pour qu'il renvoie aussi les indices de début et de fin d'un bloc non vide de somme maximale. La complexité est-elle la même ?