Aix*Marseille université

Année 2018-19

Domaine Sciences et Technologies
Licence 3 Informatique

Algorithmique 2 : TD 4
Code UE : SIN5U03C

Programmation dynamique

Exercice 1 (Le ramasseur de patates)

Un champ de patates rectangulaire est subdivisé en cellules (parcelles) sous la forme d'un échiquier de dimensions $n \times m$. Dans chaque cellule (i, j) se trouve $p(i, j) \geq 0$ pommes de terre. Un ramasseur R commence dans la cellule $(1, 1)$ et termine dans la cellule (n, m) en ramassant toutes les patates qui se trouvent sur un chemin reliant ces cellules ; notamment, si R est arrivé dans la cellule (i, j), alors il ramasse les patates de cette cellule et continue soit dans la cellule en-dessous $(i+1, j)$ soit dans la cellule de droite $(i, j + 1)$. Il ne peut jamais remonter ou aller vers la gauche. Dans l'exemple ci-contre, le ramasseur a récolté au total 82 pommes de terres en suivant le trajet grisé entre la parcelle (1, 1) et la parcelle (8, 6). Trouver un algorithme qui, étant donné un champ représenté par un tableau à deux dimensions, calcule le chemin maximisant le nombre de patates ramassées. Donner son pseudocode et analyser sa complexité. Trouver le chemin optimal pour l'exemple ci-contre.

	1	2	3	4	5	6
1	5	3	4	10	8	1
2	2	4	10	3	8	5
3	5	10	5	8	7	2
4	7	7	12	3	6	9
5	6	5	13	3	3	7
6	1	6	7	8	2	8
7	1	10	3	4	2	7
8	2	2	2	2	3	3

Exercice 2 (Distance d'édition)

Nous définissons les opérations d'insertion, suppression et substitution sur les mots d'un alphabet Λ. Pour un mot $w = w_1 w_2 \dots w_n$:

- $w' = w_1 \dots w_{i-1} w_{i+1} \dots w_n$ est obtenu depuis w par suppression de sa i-ième lettre ($i \in [1, n]$).
- $w' = w_1 \dots w_{i-1} \lambda w_i \dots w_n$ est obtenu par insertion de la lettre $\lambda \in \Lambda$ en i-ième position ($i \in [1, n + 1]$).
- $w' = w_1 \dots w_{i-1} \lambda w_{i+1} \dots w_n$ est obtenu par substitution de la i-ième lettre par la lettre $\lambda \in \Lambda$ ($i \in [1, n]$).

La *distance d'édition* entre deux mots est définie par le nombre minimum d'opérations successives pour passer d'un mot à l'autre. Par exemple, la distance entre `marseille` et `massilia` est de 4 :

- `marseille`
- `masseille` (substitution 3e lettre)
- `massille` (suppression 5e lettre)
- `massilie` (substitution 7e lettre)
- `massilia` (substitution 8e lettre)

Exprimer la distance d'édition entre deux mots par une formule récursive. Écrire un algorithme à partir de cette formule et analyser sa complexité. Utiliser cet algorithme pour calculer la distance d'édition entre `Boule` et `Bill`.

Exercice 3 (Les éléphants)

Lors d'un défilé d'une troupe de cirque passent successivement plusieurs éléphants. Chaque éléphant est caractérisé par une masse et une stupidité, codées par deux entiers. Nous voulons trouver une suite d'éléphants de plus en plus gros et de plus en plus stupides, qui respectent l'ordre du défilé mais qui ne sont pas forcément consécutifs (appelons *valide* une telle suite).

Formellement, nous disposons d'une variable nbrÉléphants codant le nombre d'éléphants, et de deux tableaux masse et stupidité, de sorte que masse$[i]$ et stupidité$[i]$ sont la masse et la stupidité du i^e éléphant ayant défilé.

(a) Donner une formule de récurrence permettant de décrire la plus longue suite valide d'éléphants d'un défilé.

(b) Écrire l'algorithme correspondant en pseudocode.

(c) Quelle est la complexité de cet algorithme ?

Exercice 4 (Le plus grand sous-carré monochromatique) Étant donné un tableau M de dimension $n \times m$ dont les entrées sont des couleurs (représentées par des entiers), un *sous-carré monochromatique de côté k* est donné par $i \leq n - k$ et $j \leq m - k$ tels que tous les $M[a, b]$ sont égaux pour $i \leq a < i + k$ et $j \leq b < j + k$. Écrire le pseudo-code d'une procédure en $O(n \cdot m)$ qui, étant donné M, renvoie un plus grand sous-carré monochromatique. Estimer sa complexité.

Exercice 5 (Playlist)

Vous vous retrouvez DJ d'une soirée étudiante. Vous avez à votre disposition une liste de n titres musicaux, indicés de 1 à n, parmi lesquels vous souhaitez en sélectionner k pour toute la soirée. Malheureusement le matériel dont vous disposez est défectueux, et vous êtes forcés de parcourir la liste dans l'ordre : après avoir diffusé le titre i, vous ne pourrez plus diffuser les titres $1, 2, \ldots, i$.

Il est important pour un DJ de ne pas enchaîner deux titres de façon disharmonieuse. Cela dépend essentiellement des morceaux choisis pour se succéder. Ainsi, si vous diffusez le titre i suivi du titre j, vous marquez $\mathsf{harmonie}[i, j]$ points. Votre but est de maximiser votre score.

(a) Écrire un algorithme qui, étant donné le tableau $\mathsf{harmonie}[1..n][1..n]$, calcule la séquence de k musiques optimisant la somme des scores marqués. Analyser sa complexité en fonction de k et n.

(b) Formuler le problème comme un problème de plus long chemin dans un graphe acyclique.