Aix*Marseille université

Année 2018-19

Domaine Sciences et Technologies
Licence 3 Informatique

Algorithmique 2 : TD 5
Code UE : SIN5U03C

Programmation dynamique II

Exercice 1 (Recherche d'un palindrome)

Un *palindrome* est un mot qui se lit indifféremment de gauche à droite ou de droite à gauche, comme par exemple `kayak` ou `ressasser` (de longueurs impaires) ou bien `serres` (de longueur paire). Une *sous-séquence* d'un mot $w_1 w_2 \cdots w_n$ est un mot $w_{i_1} w_{i_2} \cdots w_{i_k}$ avec $i_1 < i_2 < \cdots < i_k$. Par exemple `eavae` (ou bien `aimia`) est la plus longue sous-séquence palindromique de `algorithmiqueavancée`. Les *chaînes* sont des sous-séquences particulières, où l'on sélectionne des lettres consécutives dans le texte. Par exemple, la plus longue chaîne palindromique de `algorithmiqueavancée` est `ava`.

Nous voulons trouver la plus longue chaîne ou la plus longue sous-séquence d'un mot quelconque qui soit un palindrome. Afin de ne pas s'ennuyer des détails techniques, nous supposerons que le mot où chercher le palindrome est donné par un tableau d'entiers texte$[1..n]$, chaque entier codant un caractère.

(a) Décrire une méthode en $O(n^2)$ pour trouver la plus longue *chaîne* texte$[i..j]$ qui soit un palindrome (Donner votre réponse sous la forme d'une description littérale ou bien un pseudocode, et justifier sa complexité). Il peut être utile de considérer les cas des palindromes de longueurs paire ou impaire séparément.

(b) Passons au cas des sous-séquences palindromiques. Notons meilleurPalindrome$[i, j]$ le plus long palindrome *sous-séquence* de texte$[i..j]$. Donner une formule récursive décrivant meilleurPalindrome.

(c) En déduire un algorithme calculant la plus longue sous-séquence palindrome de texte. Donner votre réponse sous la forme d'un pseudocode.

(d) Quelle est la complexité de cet algorithme ?

Exercice 2 (Un jeu de rôle)

Le guerrier. Vous jouez à un jeu de rôle, dans lequel votre personnage, un guerrier cupide, affronte divers monstres. Le jeu se déroule sur n jours. Lors du jour i, le guerrier peut :

- ou bien affronter le monstre M_i, qui possède une force $f_i \in \mathbb{N}$. Dans ce cas, le guerrier perd f_i points de vie, mais il vainc le monstre et gagne p_i pièces d'or.
- ou bien se reposer, le guerrier gagne 5 points de vie (ou moins s'il atteint son maximum de vie).

Le guerrier commence désargenté avec $s = 10$ points de vie et ne peut en aucun cas avoir plus de points de vie. S'il descend à 0 point de vie ou moins, il meurt.

Donner un algorithme de complexité $O(s \cdot n)$ qui, étant donné s, deux tableaux f et p et leur longueur n, calcule la meilleure stratégie à adopter par le guerrier pour qu'il survive et soit le plus riche possible à la fin du jour n. Pour trouver cet algorithme, il vous faudra considérer, pour chaque jour i et chaque nombre de points de vie $s' \leq s$, quelle est la somme maximale d'or $t[i, s']$ que le guerrier peut avoir à la fin du jour i, de sorte qu'il lui reste s' points de vie. Trouver ensuite une formule de récurrence sur les valeurs de t, et en déduire l'algorithme.

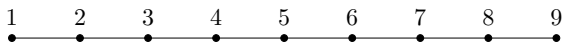
Le magicien. Vous décidez d'essayer une autre classe de personnage, en jouant un magicien sans scrupule. Les magiciens sont des êtres frêles, qui meurent au premier coup porté. On ne tient donc pas compte de la santé du magicien. Par contre, il dispose de sorts puissants qui le protègent des blessures. Mais ses sorts ne durent pas plus de trois jours, il lui faut donc prendre un jour de repos pour les ré-activer, ce qui en plus lui coûte 3 pièces d'or en achat d'ingrédients magiques. Il peut aussi se reposer gratuitement, auquel cas ses sorts ne sont pas ré-activés. Ainsi, un magicien ne peut pas choisir d'affronter les monstres trois jours de suite. Par exemple, il peut choisir de se reposer les 2e, 5e, 9e (gratuitement) et 10e jours, pour vaincre les monstres des autres jours jusqu'au 13e jour compris. Cela lui coûterait 9 pièces d'or. On considère qu'il a lancé ses sorts gratuitement la veille du premier jour.

De nouveau, donner un algorithme calculant une stratégie pour que le magicien soit le plus riche possible à la fin des n jours. Cet algorithme prend en argument n et un tableau p des gains par monstre. Quelle est sa complexité ?

Exercice 3 (Ensemble indépendant)

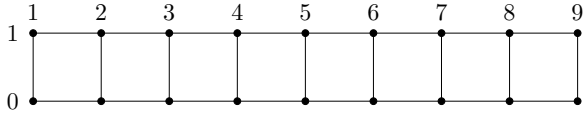
Un ensemble indépendant dans un graphe $G = (V, E)$ est un ensemble de sommets $S \subseteq V$, non-adjacents deux-à-deux : pour tous $u \in S$ et $v \in S$, u et v ne sont pas adjacents ($(u, v) \notin E$). En général, trouver un ensemble indépendant le plus grand possible est un problème algorithmiquement difficile, mais dans certains graphes on peut le résoudre.

(a) Soit P_n le graphe chemin ayant n sommets, les entiers de 1 à n, et $n-1$ arêtes de la forme $(i, i+1)$ avec $i \in \{1, 2, \ldots, n-1\}$. Voici par exemple le chemin P_9 :



On donne un poids à chaque sommet. On suppose les poids donnés dans un tableau `weight[1..n]`. Décrire un algorithme déterminant l'ensemble indépendant dont la somme des poids est maximum. Donner son pseudocode et analyser sa complexité.

(b) On considère maintenant l'échelle de longueur n dont voici un exemple :



De nouveau, on donne un poids à chaque sommet, sous la forme d'un tableau à deux dimensions `weight[0..1][1..n]`. Décrire un algorithme déterminant l'ensemble indépendant de poids maximum, donner son pseudocode et analyser sa complexité. (Indication : dans chaque colonne, l'ensemble indépendant maximum contient soit l'élément du haut, soit l'élément du bas, soit aucun des deux).

Exercice 4 (Calcul de sous-ensembles d'intervalles colorés)

Soit $I = \langle (g_1, d_1, c_1), \ldots, (g_n, d_n, c_n) \rangle$ un ensemble ordonné de n intervalles définis par leurs bornes et leurs couleurs : le i-ième intervalle de I a pour borne gauche g_i, pour borne droite d_i et sa couleur est c_i. Les intervalles dans I sont ordonnés dans l'ordre croissant de leurs bornes droites. La *longueur* d'une sous-suite d'intervalles disjoints est la somme des longueurs de ses intervalles.

Le problème qui nous intéresse est de trouver une sous-suite d'intervalles disjoints de I dans laquelle les intervalles sont ordonnés dans l'ordre croissant de leurs bornes droites, deux intervalles *successifs* sont toujours de couleurs différentes, et la longueur de cette sous-suite d'intervalles est maximale. Par exemple si I vaut $\langle (2, 4, \mathit{bleu}), (1, 5, \mathit{blanc}), (4, 6, \mathit{blanc}), (5, 10, \mathit{blanc}), (8, 11, \mathit{noir}), (10, 11, \mathit{bleu}) \rangle$, la suite $\langle (2, 4, \mathit{bleu}), (5, 10, \mathit{blanc}), (10, 11, \mathit{bleu}) \rangle$ est une sous-suite de longueur maximale (sa longueur est 8). La sous-suite $\langle (1, 5, \mathit{blanc}), (5, 10, \mathit{blanc}), (10, 11, \mathit{bleu}) \rangle$ est de longueur supérieure (sa longueur est 10), mais elle ne satisfait pas la contrainte sur les couleurs des intervalles successifs.

On note $L(i)$ la longueur maximale d'une sous-suite d'intervalles disjoints de $\{(g_1, d_1, c_1), \ldots, (g_i, d_i, c_i)\}$ qui contient l'intervalle (g_i, d_i, c_i) (pour le moment, on prend pas en compte le fait que deux intervalles successifs sont toujours de couleurs différentes).

(a) Dans le pire des cas quel est le nombre maximum de suites d'intervalles disjoints de I ?

(b) Quelles sont les valeurs $L(1), \ldots, L(5)$ avec l'exemple défini plus haut ?

(c) Définir $L(i)$ en fonction de $L(1), \ldots, L(i-1)$.

(d) Modifier la definition de $L(i)$ pour prendre en compte la contrainte que deux intervalles successifs sont toujours de couleurs différentes. Définir $L(i)$ en fonction de $L(1), \ldots, L(i-1)$. Avec cette modification, écrire une fonction qui calcule les valeurs $L(1), L(2), \ldots, L(n)$ avec une approche de type programmation dynamique (utiliser un tableau pour stocker ces valeurs).