Algorithmique et Programmation

TD n° 2 : Programmation Dynamique

École normale supérieure – Département d'informatique
algoL3@di.ens.fr

2015-2016

Exercice 1. Multiplication de matrices

Donner un algorithme en temps $O(n^3)$ qui trouve la meilleur façon pour multiplier n matrices $M_1, \ldots, M_n$ de taille $a_1 \times b_1, \ldots, a_n \times b_n$ en multipliant des matrices deux à deux. La sortie est un parenthésage de $M_1 \ldots M_n$.

Supposer qu'il faut un temps $f(m, n, p)$ pour multiplier une matrice $m \times n$ par une matrice $n \times p$ et que l'évaluation de $f(m, n, p)$ se fait en temps $O(1)$. Supposer aussi que le produit $M_1 \times \cdots \times M_n$ est bien définit.

Exercice 2. Modification de mots

La distance entre deux mots est le nombre minimum d'opérations nécessaires pour transformer un mot en l'autre, où une opération est une insertion, une suppression ou le remplacement d'une lettre par une autre.

1. Proposer un algorithme qui calcule la distance entre deux mots.
2. Proposer un algorithme qui utilise seulement une mémoire de taille $O(m + n)$.
3. Proposer un algorithme qui retourne aussi les modifications à faire pour passer d'un mot à l'autre.
4. Proposer un algorithme en temps $O(mn)$ qui utilise seulement une mémoire de taille $O(m + n)$ et qui retourne aussi les modifications à faire.

Exercice 3. Somme et produit maximum

Pour cet exercice, nous supposons que toute opération arithmétique (même la multiplication) se fait en $O(1)$.

1. Donner un algorithme polynomial qui prends en entrée un tableau $A[0 \ldots n-1]$ de taille n et qui revoie la plus grande somme possible d'un sous-tableau contigu $A[i \ldots j]$.
2. Donner un algorithme en temps $O(n)$ pour la question précédente.
3. Donner un algorithme en temps $O(n)$ qui prends en entrée un tableau $A[0 \ldots n-1]$ et qui revoie le plus grand produit possible d'un sous-tableau contigu $A[i \ldots j]$.
   (Supposer que le sous-tableau vide a un produit de 1.)

Exercice 4. CHOMP

Dans le jeu de *Chomp*, deux joueurs mangent en alternance les carrés d'un tablette de chocolat $m \times n$. Le but est d'éviter le coin en bas, à gauche qui est empoisonnée. À chaque tour, un joueur choisis un carré et mange celui-ci et tous les carrés en haut et à droite de celui-ci.

```
000000          000
000000  (3, 2)  000
000000  -----> 000
000000          000000
```

Donner un algorithme avec un précalcul en temps $(m+n)^{m+n}$ pour jouer à ce jeu. C'est-à-dire, après un calcul initial, à chaque fois que c'est son tour, cet algorithme retourne en $O(\min(m, n) \log(mn))$ un coup à jouer qui gagne à coup sûr ou indique qu'un tel coup n'existe pas. L'algorithme reçoit ensuite le coup de l'adversaire comme une paire de coordonnées.