Programmation dynamique

1 Multiplication matricielle

Soit n un entier naturel supérieur ou égal à 3. On considère une famille de n matrices M_0, M_1, …, M_{n-1} d'entiers et on suppose que leurs tailles permettent le calcul du produit $M_0 \times M_1 \times \cdots \times M_{n-1}$. On veut calculer ce produit en minimisant le nombre de multiplications entre entiers réalisées.
Dans l'exercice, les matrices sont représentées par des listes de listes.

1. Écrire une fonction `produit` qui prend en argument deux matrices et calcule leur produit lorsque leurs tailles sont compatibles. Calculer la complexité de cette fonction en nombre de multiplications d'entiers.
2. Soit $p \in \mathbb{N}$ tel que $p \geq 3$. Soient trois matrices $M_0 \in \mathcal{M}_{2,p}(\mathbb{Z})$, $M_1 \in \mathcal{M}_{p,2}(\mathbb{Z})$ et $M_2 \in \mathcal{M}_{2,p}(\mathbb{Z})$. Déterminer les nombres de multiplications entre entiers effectuées lorsqu'on calcule le produit $M_0M_1M_2$ en faisant :
   - $M_0(M_1M_2)$;
   - $(M_0M_1)M_2$.

   Commenter le résultat obtenu.

Par associativité de la multiplication matricielle, tous les parenthésages de M_0, M_1, …, M_{n-1} lors du calcul du produit donnent le même résultat. Mais le nombre de multiplications d'entiers dépend du parenthésage effectué. On cherche un parenthésage qui minimise ce nombre de multiplications.
On suppose que les tailles des matrices M_0, M_1, …, M_{n-1} sont stockées dans une liste L de la façon suivante : pour tout $i \in [\![0, n-1]\!]$, la matrice M_i admet $L[i]$ lignes et $L[i+1]$ colonnes.

3. En notant $p(i,j)$ le nombre minimal de multiplications d'entiers nécessaire au calcul du produit $M_i \times \cdots \times M_j$, justifier la sous-structure optimale du problème.
4. Écrire une fonction `min_mul` qui prend en paramètre une liste de tailles de matrices et qui renvoie le nombre minimal d'opération nécessaires au calcul du produit matriciel correspondant.

   On proposera une version ascendante et une version descendante.
5. Écrire une fonction `mul_opt` qui prend en paramètre une liste de tailles de matrices et qui renvoie un parenthésage optimal. Par exemple, pour des matrices comme celles de la question 3, la fonction doit renvoyer `'(M0.M1).M2'`. correspondant.

   On proposera une version ascendante et une version descendante.

2 Ordonnancement des tâches pondérées

On considère un séminaire comportant plusieurs conférences représenté par une liste S constituée de listes à trois éléments. Chacune de ces listes contient le début, la fin et la valeur attribuée à la conférence.
En notant $S = [[d_0, f_0, v_0], \cdots, [d_{n-1}, f_{n-1}, v_{n-1}]]$, on supposera que la liste $[f_0, \cdots, f_{n-1}]$ est triée dans l'ordre croissant
On cherche à maximiser le séminaire avec la contrainte de n'assister qu'à des conférences en entier mais en supposant que l'on peut passer instantanément de l'une à l'autre, c'est-à-dire que l'on peut à la fin de la conférence d'indice i se rendre à celle d'indice j dès que $f_i \leq d_j$.

1. Justifier que ce problème a une sous-structure optimale.
2. Écrire une fonction `max_semi` qui prend en paramètre une liste représentant un séminaire et renvoie la valeur maximale de celui-ci.

   On proposera une version ascendante et une version descendante.
3. Écrire une fonction `org_semi` qui prend en paramètre une liste représentant un séminaire et renvoie une liste des indices des conférences auxquelles il faut assister pour maximiser sa venue au séminaire.

   On proposera une version ascendante et une version descendante.