Algorithmique Avancée
exercices

Exercice 1 (Produits de Matrices en Chaîne)

Soient A, B et C trois matrices telles que $C = A \times B$. Notons leurs dimensions respectives $m \times n$, $n \times p$ et $m \times p$. Par définition du produit des matrices $C(i,j) = \sum_{k=1}^{n} A(i,k) * B(k,j)$.

1. *Ecrivez (en pseudocode) un algorithme pour calculer C en utilisant directement la formule ci-dessus.*
   Montrer que le nombre de multiplications nécessaires à l'exécution de cet algorithme est $m * n * p$.
2. *Soit $M_1 \times M_2 \times ... \times M_r$ une chaîne de produit de matrices. On a vu que l'on peut calculer cette expression de nombreuses manières. Par exemple : $(...(M_1 \times M_2) \times ... \times M_r)$ ou $(M_1 \times (M_2 \times (... \times M_r))...)$.*
   En cours nous avons considéré que le coût de calcul de $M_1 \times M_2 \times ... \times M_r$ est le nombre de multiplications utilisées.
   Considérons le cas $r = 4$ où les matrice M_1 à M_4 sont de dimensions 100×1, 1×100, 100×1 *et* 1×100. *Il y a cinq manières de calculer le produit $M_1 \times M_2 \times M_3 \times M_4$. Combien coûte chacune de ces manière ?*
3. *Comme au cours notons $M_{i,j}$ le produit $M_i \times M_{i+1} \times ... \times M_j$ et $M_{i,i} = M_i$, avec $1 \leq i,j \leq r$.*
   Définissons $S = \{p_1, p_2, ..., p_{r-1}\}$ comme une <u>séquence de produits</u> ssi chaque produit p_i est de la forme $M_{i,j} \times M_{j+1,q}$ et que $M_{i,j}$ et $M_{j+1,q}$ ont été calculés lors d'une étape précédentes, c'est-à-dire lors d'un p_l avec $l < k$, ou sont de la forme $M_{t,t}$.
   Remarquez au passage que le résultat de p_k est $M_{i,q}$, et que pour calculer correctement le produit $M_1 \times M_2 \times ... \times M_r$ on doit nécessairement suivre une séquence de produits.
   On dira que deux séquences de produits $S_1 = \{p_1, ..., p_{r-1}\}$ et $S_2 = \{q_1, ...q_{r-1}\}$ sont <u>différentes</u> si $p_k \neq q_k$ pour un k donné.
   Montrer qu'il y a $(r-1)!$ séquences de produits différentes.
4. *D'un point de vue purement analytique, si on considère l'ordre des multiplications entre matrices, il y a $(r-1)!$ séquences de produits différentes, mais de nombreuses séquences sont essentiellement identiques.*
   Par exemple, les séquences $S_1 = \{M_{1,2} = M_1 \times M_2, M_{3,4} = M_3 \times M_4, M_{1,2} \times M_{3,4}\}$ et $S_2 = \{M_{3,4} = M_3 \times M_4, M_{1,2} = M_1 \times M_2, M_{1,2} \times M_{3,4}\}$ ne diffèrent que par l'ordre des multiplications ; tous les résultats intermédiaires, les $M_{i,j}$ calculés, sont identiques.
   Nous dirons que deux séquences de produits S_1 et S_2 sont <u>essentiellement distinctes</u> si au moins un résultat intermédiare de S_1 n'est pas un résultat intermédiaire de S_2. Montrer

que le nombre de séquences de produits essentiellement distinctes est égal au nombre d'arbres binaires de $r-1$ noeuds exactement.

5. *Montrer que le nombre d'arbres binaires de n noeuds exactement est $\frac{1}{n+1}\binom{2n}{n}$.*

Exercice 2 (Produits de Matrices en Chaîne)

Dans l'exercice précédent, on a vu que le nombre de manière différentes d'évaluer une chaîne de produits de matrices est très grand même si r est relativement petit (disons 10 ou 20). Dans le cours on a développé un algorithme en $O(r^3)$ pour trouver une séquence de produits optimale (la moins coûteuse en terme de multiplications).

1. *Programmez en Java ou en C++ l'algorithme présenté en cours.*
   Verifiez son fonctionnement avec l'exemple donné en cours.
2. *Quelle est la meilleure manière de calculer l'exemple de l'exercice précédent ? Combien coûte-t-elle ?*
3. *Modifier (sur papier) le code vu en cours de manière à obtenir non seulement le nombre minimal, mais surtout l'ordre des opérations à effectuer.*
   (Il s'agit principalement de placer des parenthèses au bons endroits.)
4. *Implémentez cette modification, par exemple en C++ ou en Java.*

Exercice 3

On considère un alphabet Σ et un langage L sur l'alphabet Σ, défini par une grammaire donnée par un ensemble de règles de productions :
- *$\Sigma = \{s_1, \ldots, s_m\}$ est un ensemble fini de symboles.*
- *$R = \{R_1, ..., R_p\}$ est un ensemble de règles de productions du type $R_j = [s_{i_1} \to s_{i_2}s_{i_3}]$ ($1 \le j \le p$ et $\{i_1, i_2, i_3\} \subseteq \{s_1, ..., s_m\}$).*

  Autrement dit, on a R une grammaire permettant de générer des mots sur l' alphabet Σ.

Définition:

Soient x et y deux mots sur l' alphabet Σ. On dit que x peut être transformé en y, si $x = y$ ou si un nombre fini d' applications de règles de productions permettent de transformer x en y.

Notation:

$x \to^* y$ signifie que x peut être transformé y.

Le but de cet exercice est d'utiliser la programmation dynamique pour obtenir un algorithme qui détermine si x peut être transformé en y.

Définition:

Soit $x = x_1x_2...x_n \in \Sigma^n$. On définit $P(i,j,k) = 1$ si $s_k \to^* x_i \cdots x_{i+j-1}$, et $P(i,j,k) = 0$ sinon. [1]

Remarque:

1. *On a la condition initiale suivante : $P(i,1,k) = 1$ ssi $x_i = s_k$.*

[1] En d' autres termes, $P(i,j,k) = 1$ quand la $k$ème lettre de l' alphabet Σ peut être transformée en le sous-mot de x de longueur j, en commençant à la $i$ème position de x.

2. *On a la relation de récurrence suivante :* $P(i,j,k) = 1$ *si et seulement si la conjonction des deux assertions suivantes tient :*
   - $\exists k, k' \in \{1, \ldots, m\}$ *tel que* $s_k \rightarrow^* s_{k'} s_{k''}$
   - $\exists 1 \leq j' < j$ *tel que* $P(i, j', k') \wedge P(i + j', j - j', k'')$
3. *On a la condition de terminaison suivante :* $s_k \rightarrow^* x$ *ssi* $P(1, n, k) = 1$.

Il vous est demandé de faire les exercices suivants :

1. *En vous inspirant de la remarque ci-dessus, utiliser la technique de la programmation dynamique pour construire un algorithme permettant de décider si* $s_k \rightarrow^* x$, *(pour* $x \in \Sigma^*$ *et* $s_k \in \Sigma$ *donnés).*
2. *De même, proposer un algorithme décidant si* $x \rightarrow^* y$, *pour* $x, y \in \Sigma^*$.
3. *Analyser la complexité en temps des deux algorithmes précédants, et répondre à la question suivante :*

   > *Est-il possible de résoudre le premier des deux problèmes ci-dessus en temps* $\mathcal{O}(n^3)$ *? Si oui, donnez un algorithme. Sinon, prouvez votre affirmation.*

Remarque:

Ce problème est extrêmement important, puisqu' il traite de la question de savoir comment décider de l' appartenance d' un mot à un langage hors-contexte sous forme normale CNF (Chomsky normal form).