Dea Algorithmique

Paradigmes de la combinatoire et de l'algorithmique

Premier problème d'entraînement

Septembre 2002

Les sous-mots

On ecrira les algorithmes demandés en utilisant un des langages de programmation de votre choix: C, Java, Pascal, Caml.

I. Généralités

On dit que le mot u est un sous mot du mot f s'il s'obtient en supprimant certaines lettres de f. Par convention le mot vide est un sous-mot de f quelque soit f (suppression de toutes les lettres) et f est sous-mot de lui même.

Par exemple *abba* est sous-mot de *cababcbbbaaaab*, noter qu'un sous-mot n'est pas nécessairement un facteur.

Question 1. Donner un algorithme qui teste si le mot u de longeur k est sous-mot du mot f de longueur n. On notera `f[i]` la i-ème lettre de f $(1 \leq i \leq n)$ et `u[j]` la j-ème lettre de u $(1 \leq j \leq k)$

Question 2. Combien un mot de longueur k peut-il avoir de sous-mots différents au maximum? Et au minimum? Proposez deux exemples de mots pour lequels ce minimum et ce maximum sont atteints.

Question 3. On dit que deux mots f et g sont incomparables si aucun n'est sous-mot de l'autre. Quel est le nombre maximum de sous-mots incomparables de *abab*? Quel est le nombre maximum de sous-mots incomparables d'un mot f de longueur n?

Question 4. On note $\pi_{a,b}(f)$ le sous-mot de f qui contient toutes les occurrences de a et de b. Par exemple $\pi_{a,c}(cababcbbbaaaab) = caacaaaa$. Montrer que si A est un alphabet de taille k, alors la connaissance de $\pi_{a,b}(f)$ pour toutes les $\frac{k(k-1)}{2}$ paires de lettres $\{a, b\}$ de A permet de déterminer f de manière unique. Peut-on se passer de l'un des $\pi_{a,b}(f)$ pour effectuer cette détermination?

II. Coefficients binomiaux

Dans cette partie on note $\binom{f}{u}$ le nombre de façons d'obtenir u en supprimant des lettres de f. Par exemple:

$$\binom{abbbaa}{ba} = 6$$

Question 1. Dans la suite u^k représente le mot obtenu en concaténant k copies de u. Quelles sont les valeurs de

$$\binom{a^p b^q a^r}{ab}, \quad \binom{a^p b^q a^r}{aba}, \quad \binom{(ab)^p}{ab} \ ?$$

Question 2. Démontrer la formule suivante, dans laquelle f, g sont deux mots et a, b deux lettres ($\delta_{a,b}$ vaut 1 si $a = b$ et 0 sinon):

$$\binom{fa}{gb} = \binom{fa}{gb} + \delta_{a,b}\binom{fa}{gb}$$

Question 3. Montrer que la construction du triangle de Pascal est un cas particulier de cette formule.

Question 4. Donner un algorithme récursif de calcul de $\binom{f}{g}$ qui utilise cette formule. Quel est le nombre d'appels de fonctions qui est effectué par cet algorithme en fonction des longueurs n et m respectivement de f et de g? Donner un algorithme itératif qui effectue le calcul en utilisant un tableau de n lignes et m colonnes.

Question 5. Démontrer les formules suivantes:

$$\binom{fg}{h} = \sum_{uv=h} \binom{f}{u}\binom{g}{v}$$

$$\sum_{h \in A^*} (-1)^{|h|} \binom{f}{h}\binom{h}{g} = (-1)^{|f|} \delta_{f,g}$$

III. Plus longs sous-mots communs

On se propose de déterminer pour deux mots f et g un mot u qui soit à la fois sous-mot de f et de g et qui soit le plus long parmi les mots qui satisfont cette propriété.

Dans la suite on note $f[i]$ la i-ème lettre de f et f_i le facteur composé des lettres de f entre la première et la i-ème. Ainsi si $f = abbbabba$, $f[5] = a$ et $f_5 = abbba$. On désigne par n la longueur de f et m celle de g.

On note $\lambda(f, g)$ la longueur du plus long sous-mot commun à f et g.

Question 1. Montrer que pour deux lettres a et b

$\lambda(fa, gb)$ s'exprime à l'aide de $\lambda(f, gb), \lambda(fa, g)$ et $\lambda(f, g)$.

Question 2. En déduire un algorithme qui calcule le tableau `lambda[i][j]` des longueurs des plus longs sous-mots communs à f_i et g_j, pour i variant entre 1 et la longueur n de f et j variant entre 1 et la longueur m de g. Pour simplifier on posera:

`lambda[0][j] = lambda[i][0] = 0,` pour $0 \leq i \leq n, 0 \ \leq j \leq m$

.

Question 3. Construire le tableau pour les mots $f = abbabbba$ et $g = aaabbab$. Utiliser le tableau ainsi calculé pour obtenir un plus long sous-mot commun à ces deux mots.

Question 4. Donner un algorithme qui calcule un plus long sous-mot commun en utilisant le tableau `lambda[i][j]`.

Question 5. Proposer un nouvel algorithme de calcul de la longueur du plus long sous-mot commun ayant la même complexité en temps que le précédent et qui n'utilise que deux tableaux d'une seule ligne de longueur $min(n, m)$ au lieu du tableau `lambda[i][j]` à deux entrées.

On se propose de donner un algorithme de calcul du plus long sous-mot commun qui fonctionne en temps $O(mn)$ et en espace $O(min(n, m))$. Pour cela on utilise la méthode diviser pour régner et on se propose de commencer par calculer le plus long sous-mot commun à f_p et g_k où p est égal à la partie entière de $n/2$ et k un entier à déterminer. Celui-ci doit être tel que

$$\lambda(f,g) = \lambda(f_p, g_k) + \lambda(f'_p, g'_k) \text{ avec } f = f_p f'_p, \quad g = g_k g'_k$$

Question 6. Pour déterminer l'entier k, on calcule pour chaque couple i, j tels que $1 \leq i \leq n$ et $q \leq j \leq m$ l'entier $\ell = \mu(i,j)$ satisfaisant:

$$\lambda(f_i, g_j) = \lambda(f_\ell, g_q) + \lambda(f''_\ell, g''_j)$$

avec $f_i = f_\ell f''_\ell, g_j = g_q g''_j$.

Exprimer $\mu(i,j)$ en fonction de $\mu(i-1,j),\ \mu(i,j-1),\ \mu(i-1,j-1)$.

Question 7. Donner un algorithme de calcul de plus long sous-mot commun fonctionnant en temps $O(mn)$ et en espace $O(min(n,m))$.