Algorithmique

Partiel du 15 novembre 2012

durée 3 heures

Le polycopié du cours est le seul document autorisé.

Les exercices sont indépendants.

Les procédures et fonctions devront être écrites en pseudo-code (pas en Java, Caml, ...)

Toutes les réponses devront être correctement justifiées. La rigueur des raisonnements, la clarté des explications, et la qualité de la présentation influeront sensiblement sur la note.

Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Preuve et terminaison

On considère le programme suivant (toutes les variables sont entières) :

```
Lire a et b
p ← 3; d ← 1; m ← 1
tant que a > 1 ou b > 1 faire
   cas
      p divise a et p divise b: a := a/p; b ← b/p; d ← d · p; m ← m · p
      p divise a et p ne divise pas b: a ← a/p; m ← m · p
      p ne divise pas a et p divise b: b ← b/p; m ← m · p
      p ne divise pas a et p ne divise pas b: p ← p + 2
   fincas
ftq
Afficher d et m
```

On suppose que les données a et b sont deux entiers **impairs positifs**.

[5] **a)** Qu'affiche ce programme sur les données $a = 45$ et $b = 75$?
Montrer que ce programme termine.
Que calcule ce programme ?
Prouver avec la méthode de Hoare que ce programme calcule bien cela.

2 Complexité

On considère le programme suivant ($t[1..n]$ est un tableau d'entiers strictement positifs) :

$m \leftarrow 0$; $j \leftarrow 0$;
pour $i \leftarrow 1$ **à** n **faire**
  si $t[i] > m$ **alors** $m \leftarrow t[i]$; $j \leftarrow i$ **fsi**
fpour

Pour un entier $k \in \{1, \ldots, n\}$, on note $X_k \colon \mathfrak{S}_n \to \{0, 1\}$ la variable aléatoire définie par $X_k(\sigma) = 1$ si et seulement si l'instruction $m \leftarrow t[k]$ est exécutée par le programme ci-dessus lorsque le tableau t est initialisé avec σ.

[2] **a)** Montrer que $E(X_k) = \frac{1}{k}$.

Soit $X \colon \mathfrak{S}_n \to \{1, \ldots, n\}$ la variable aléatoire telle que $X(\sigma)$ est le nombre de fois que l'instruction $m \leftarrow t[i]$ est exécutée par le programme ci-dessus lorsque le tableau t est initialisé avec σ.

[2] **b)** Montrer que $E(X) \leq 1 + \ln(n)$.

3 Alignement de séquences

Il s'agit d'aligner *au mieux* deux mots. L'application principale est l'alignement de séquences du génome, représentées par des mots sur l'alphabet $\{A, T, C, G\}$ pour l'ADN ou $\{A, U, C, G\}$ pour l'ARN. Les mots à aligner sont très longs : entre 10^3 et 10^6 lettres pour un gène.

Un autre exemple d'application est la correction automatique d'orthographe. Si on tape "ocurrance", un système intelligent (smartphone/éditeur [1]) proposera "occurrence".

Considérons deux mots $x = x_1 \cdots x_m$ et $y = y_1 \cdots y_n$ sur un alphabet Σ fixé. Un alignement est une suite de couples $(i_1, j_1)(i_2, j_2) \cdots (i_k, j_k)$ telle que $1 \leq i_1 < i_2 < \cdots < i_k \leq m$ et $1 \leq j_1 < j_2 < \cdots < j_k \leq n$. Par exemple, $A = (2,1)(3,2)(4,3)(5,4)(6,5)$ et $B = (2,1)(3,2)(5,4)(6,5)$ sont deux alignements des mots $x =$ `global` et $y =$ `local` que l'on représente par

$$A = \begin{pmatrix} g & l & o & b & a & l \\ - & l & o & c & a & l \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} g & l & o & b & - & a & l \\ - & l & o & - & c & a & l \end{pmatrix}.$$

La qualité d'un alignement est mesurée par un *coût* utilisant un paramètre $\delta > 0$ pour les "trous" (lettres non alignées) et des paramètres $\alpha(a, b) \geq 0$ pour les lettres alignées ($a, b \in \Sigma$). Le coût d'un alignement $C = (i_1, j_1)(i_2, j_2) \cdots (i_k, j_k)$ de deux mots $x = x_1 \cdots x_m$ et $y = y_1 \cdots y_n$ est

$$\texttt{coût}(C) = (m + n - 2k)\delta + \sum_{1 \leq \ell \leq k} \alpha(x_{i_\ell}, y_{j_\ell}).$$

En général, on suppose que $\alpha(a, a) = 0$ pour $a \in \Sigma$ (mais ce n'est pas nécessaire). Avec cette hypothèse, on obtient $\texttt{coût}(A) = \delta + \alpha(b, c)$ et $\texttt{coût}(B) = 3\delta$ pour l'exemple ci-dessus. L'alignement A est meilleur que B si $\alpha(b, c) \leq 2\delta$.

Dans la suite, les deux mots $x = x_1 \cdots x_m$ et $y = y_1 \cdots y_n$ sont fixés. On cherche un alignement optimal (i.e., ayant un coût minimal) de x et y.

1. dommage que ce ne soit pas encore disponible sur les stylos :)

Pour $0 \leq i \leq m$ et $0 \leq j \leq n$, on note $\texttt{opt-g}(i, j)$ le coût d'un alignement optimal de $x_1 \cdots x_i$ et de $y_1 \cdots y_j$.

[3] **a)** Combien vaut $\texttt{opt-g}(0, j)$ pour $0 \leq\ j \leq n$.
Donner un algorithme récursif pour calculer en temps $\mathcal{O}(mn)$ et en espace $\mathcal{O}(mn)$ toutes les valeurs $\texttt{opt-g}(i, j)$ pour $0 \leq i \leq m$ et $0 \leq j \leq n$.
Donner un algorithme qui, en supposant déjà calculées les valeurs ci-dessus, affiche un alignement optimal de x et y en temps $\mathcal{O}(m + n)$.

[2] **b)** Le problème majeur de l'algorithme ci-dessus est sa complexité en espace. Pour aligner deux séquences du génome ayant chacune environ 10^5 lettres, il faudrait 10 Go de mémoire. On veut donc un algorithme qui fonctionne en espace $\mathcal{O}(m + n)$.
Donner un algorithme itératif pour calculer le coût d'un alignement optimal pour x et y en temps $\mathcal{O}(mn)$ et en espace $\mathcal{O}(m + n)$.
Expliquer pourquoi cet algorithme ne permet plus de reconstruire et d'afficher un alignement optimal.

Pour $1 \leq i \leq m+1$ et $1 \leq j \leq n+1$, on note $\texttt{opt-d}(i, j)$ le coût d'un alignement optimal de $x_i \cdots x_m$ et de $y_j \cdots y_n$.

[2] **c)** Pour un entier $1 \leq \ell < m$ fixé, montrer que

$$\texttt{opt-g}(m, n) = \min\{\texttt{opt-g}(\ell, j) + \texttt{opt-d}(\ell + 1, j + 1) \mid 0 \leq j \leq n\}\,.$$

[2] **d)** Pour un entier $1 \leq \ell < m$ fixé, donner un algorithme pour calculer toutes les valeurs $(\texttt{opt-g}(\ell, j))_{0 \leq j \leq n}$ et $(\texttt{opt-d}(\ell + 1, j + 1))_{0 \leq j \leq n}$ en temps $\mathcal{O}(mn)$ et en espace $\mathcal{O}(m + n)$.

[4] **e)** Donner un algorithme récursif qui affiche un alignement optimal pour x et y en temps $\mathcal{O}(mn)$ et en espace $\mathcal{O}(m+n)$. Justifier la complexité en temps et la complexité en espace de votre algorithme.

On considère maintenant un mot $x = x_1 \cdots x_m$ et un ensemble $V = \{v_1, \ldots, v_k\}$ où chaque v_i est un mot de longueur au plus n (en général m est beaucoup plus grand que n). Il s'agit d'aligner x de façon optimale avec une concaténation (à déterminer) de mots de V. Le coût optimal est maintenant

$$\texttt{opt-star}(x, V) = \min\{\texttt{opt-g}(x, y) \mid y \in V^*\}$$

où V^* est l'ensemble des concaténations de mots de V, par exemple $v_2v_4v_1v_2v_7 \in V^*$ (il n'est pas nécessaire d'utiliser tous les mots de V et chaque mot de V peut être utilisé plusieurs fois).

[4] **f)** Donner un algorithme qui calcule $\texttt{opt-star}(x, V)$ et affiche un alignement optimal correspondant en temps polynomial. Préciser la complexité de votre algorithme.