Partiel Algorithmique 2ème semestre 2017/2018

Vous pouvez traiter chaque question en utilisant les résultats des questions précédentes même si vous ne les avez pas démontrés.

Partie 1

L'objectif de cette partie est de démontrer que l'algorithme 1 calcule l'automate (déterministe et complet) du motif P de longueur m.

Algorithme 1 : Une construction de l'automate du motif

```
Aut(P) : une table de transition indicée par {0,...,m} × Σ
Input : P, un motif de longueur m
Output : T la table de transition indicée par {0,...,m} × Σ
Data : i, j, k, ℓ des entiers

for a ∈ Σ do T[0, a] ← 0
for i from 1 to m do
    j ← T[i − 1, P[i]] ; T[i − 1, P[i]] ← i
    for a ∈ Σ do T[i, a] ← T[j, a]
end
return T
```

$\mathcal{A}_i$ désignera l'automate dont les états sont $\{0, \dots, i\}$, l'état initial 0, l'état final i, et la table de transition celle définie par T à la fin de la $i^{ème}$ itération pour $i > 0$ et avant le début de la première itération pour $i = 0$.

Question 1. Soit $\Sigma = \{a, b\}$. Dérouler l'algorithme 1 pour $P = aabaa$.
En particulier, on dessinera les automates $\mathcal{A}_i$ et on indiquera la valeur de j à la fin de chaque itération.

On notera $\to_i$ la fonction de transition de l'automate $\mathcal{A}_i$ étendue aux mots.

Question 2. Soit $i > 0$. Montrer que :
$\exists a \in \Sigma\ i \xrightarrow{a}_i i$ si et seulement si $a = P[i]$ et $i - 1 \xrightarrow{a}_{i-1} i - 1$.

Soit $i > 0$. On définit la fonction f_i de $\{0, \dots, i\}$ vers $\{0, \dots, i-1\}$ par $f_i(k) = k$ pour $k < i$ et $f_i(i)$ égale à la valeur de j à la fin de la $i^{ème}$ itération.

Question 3. Soit $i > 0$, $a \in \Sigma$ et $k, k' \leq i$. Montrer que :
Si $k \xrightarrow{a}_i k'$ alors $f_i(k) \xrightarrow{a}_{i-1} f_i(k')$.
On distinguera les cas suivants :

- $k < i$;
- $k = i$ et $k' < i$;
- $k = k' = i$.

Question 4. Soit $i > 0$ et $w \in \Sigma^*$. Montrer que :
Si $0 \xrightarrow{w}_{i-1} i - 1$ alors :

- soit $0 \xrightarrow{w}_i i - 1$;
- soit $0 \xrightarrow{w}_i i$ et $f_i(i) = i - 1$.

Question 5. Montrer par induction sur $i \geq 0$ que $\mathcal{A}_i$ est l'automate du motif $P[1, i]$.

Question 6. Quelle est la complexité de l'algorithme 1 ?

Partie 2

On considère dans ce problème, des matrices à coefficients booléens. Le produit de telles matrices est défini de manière standard avec le $\vee$ qui joue le rôle du $+$ et le $\wedge$ qui joue le rôle du $\times$. Dans la suite, on identifie 0 à **false** et 1 à **true**.

Etant donné un vecteur booléen $\boldsymbol{v}$ de dimension m, on définit $0 \leq val(\boldsymbol{v}) < 2^m$ par $val(\boldsymbol{v}) = \sum_{j=1}^{m} 2^{j-1}\boldsymbol{v}[j]$.

Question 7. Montrer qu'on peut calculer $val(\boldsymbol{v})$ en $\Theta(m)$ opérations entières.

Question 8. Soit un entier $0 \leq i < 2^m$. Montrer qu'on peut calculer le vecteur $val^{-1}(i)$ en $\Theta(m)$ opérations entières.

Soit une matrice $\boldsymbol{B}$ de dimension $m \times n$. On définit la matrice $\boldsymbol{H}$ de dimension $\{0, \ldots, 2^m - 1\} \times n$ par $\boldsymbol{H}[i,j] = \sum_{k=1}^{m} \boldsymbol{v}[k]\boldsymbol{B}[k,j]$ où $\boldsymbol{v} = val^{-1}(i)$.

Question 9. Proposer un algorithme qui calcule $\boldsymbol{H}$ en $\Theta(2^m n)$ opérations entières et booléennes.
Indication : On pourra introduire une fonction récursive $Compute(k)$ qui calcule les lignes de $\boldsymbol{H}$ indicées par un entier inférieur à 2^k.

Soit une matrice $\boldsymbol{A}$ de dimension $n \times m$. On définit la matrice produit $\boldsymbol{C} = \boldsymbol{A} \cdot \boldsymbol{B}$ de dimension $n \times n$.

Question 10. Quelle est la complexité du calcul de $\boldsymbol{C}$ par l'algorithme standard de multiplication en fonction de n et m ?

Question 11. Proposer un algorithme de calcul de $\boldsymbol{C}$ incluant le calcul de $\boldsymbol{H}$ qui opère en $\Theta(2^m n + n(n+m))$. On démontrera la complexité de l'algorithme.

On considère maintenant des matrices $\boldsymbol{A}$ et $\boldsymbol{B}$ de dimension $n \times n$ pour lesquelles on désire calculer leur produit $\boldsymbol{C}$. On suppose que m divise n. Pour $1 \leq k \leq \frac{n}{m}$, on définit les matrices $\boldsymbol{A}_k$ de dimension $n \times m$ composées des lignes de $\boldsymbol{A}$ d'indices i tels que $(k-1)m < i \leq km$ et $\boldsymbol{B}_k$ de dimension $m \times n$ composées des colonnes de $\boldsymbol{B}$ d'indices i tels que $(k-1)m < i \leq km$. On observe que :

$$\boldsymbol{C} = \sum_{k=1}^{\frac{n}{m}} \boldsymbol{A}_k \cdot \boldsymbol{B}_k$$

et que si m ne divise pas n, on peut tout de même utiliser cette décomposition en complétant $\boldsymbol{A}_{\lceil \frac{n}{m} \rceil}$ (resp. $\boldsymbol{B}_{\lceil \frac{n}{m} \rceil}$) par des lignes (resp. colonnes) nulles.

Question 12. En utilisant cette décomposition pour un m approprié et l'algorithme de la question 11, proposer un algorithme du produit $\boldsymbol{C}$ en $\Theta(\frac{n^3}{\log(n)})$.

Partie 3

On considère les codes (préfixes) C sur l'alphabet binaire $\Sigma = \{0, 1\}$ pour les événements $\{x_1, \ldots, x_n\}$ de *poids* $p_1 \geq \ldots \geq p_n \geq 0$ qui vérifient la contrainte : pour tout i, $l_i \leq L$ avec $l_i = |C(x_i)|$ et L une borne entière.

Question 13. Montrer qu'un tel code existe si et seulement $\log_2(n) \leq L$. On supposera dans la suite cette condition remplie.

Un code optimal C est un code qui minimise la quantité $L(C) = \sum_{i=1}^{n} l_i p_i$.

Question 14. Montrer qu'il existe un code optimal tel que pour tout $1 \leq i < n$, $l_i \leq l_{i+1}$ et $C(x_i) \leq_{lex} C(x_{i+1})$ où $\leq_{lex}$ désigne l'ordre lexicographique.

Pour $1 \leq i \leq j \leq n$ et $0 \leq m \leq L$, $L(i, j, m)$ désigne la longueur d'un code optimal pour les événements $\{x_i, \ldots, x_j\}$ tel que la longueur de tout mot du code est inférieure ou égale à m. Par convention, $L(i, i, m) = 0$ pour tout i et tout m (car le message n'apporte aucune information).

Question 15. Démontrer que pour tout $i < j$:

- $L(i, j, m) = \infty$ si $m < \log_2(j - i + 1)$;
- $L(i, j, m) = \sum_{u=i}^{j} p_u + \min(L(i, k, m-1) + L(k+1, j, m-1) \mid i \leq k < j)$ si $m \geq \log_2(j - i + 1)$.

Question 16. En déduire un algorithme de calcul d'un code optimal basé sur la programmation dynamique dont on analysera la complexité en fonction de L et de n.