CONCOURS CENTRALE•SUPÉLEC

Informatique

MP

3 heures Calculatrices autorisées

2011

Les candidats indiqueront en tête de leur copie le langage de programmation choisi (Pascal ou Caml). Les candidats ayant choisi Caml devront donner le type de chaque fonction écrite. Les candidats travaillant en Pascal pourront écrire des fonctions ou des procédures.

Les deux parties sont indépendantes.

I Identification de sous-mots répétés dans un mot

Cette partie traite de l'identification de sous-mots répétés dans un mot formé sur un alphabet A donné. Ce problème est particulièrement prégnant en génétique où il s'agit de repérer des répétitions dans des brins d'ADN, ce qui permet de comprendre sa structure. Dans ce cas, on prend simplement $A = \{g, t, a, c\}$.

Soit A un alphabet et soit $m = m_0 m_1 \dots m_{n-1}$ un mot de longueur n formé sur A ($m_i \in A$ pour tout i).

Un mot sera représenté par une chaîne de caractères. Aussi bien en Caml qu'en Pascal, on extraira d'un mot m le caractère de rang i à l'aide d'une fonction `caractère`(m, i), supposée fournie. Le rang du premier caractère est 0.

I.A – Une famille de relations d'équivalence

Soit $k \in \{1, 2, \dots, n\}$ et soient $i, j \in \{0, 1, \dots, n-k\}$. Les rangs i et j sont dits k-équivalents si

$$x_i x_{i+1} \dots x_{i+k-1} = x_j x_{j+1} \dots x_{j+k-1}$$

c'est-à-dire si les sous-mots de longueur k commençant aux indices i et j sont identiques.

Ce fait est noté $\langle i, E_k, j \rangle$.

I.A.1) Montrer que $\langle i, E_{a+b}, j \rangle$ peut se déduire de deux expressions simples construites sur E_a et E_b.

I.A.2) Montrer que $\langle i, E_{a+b}, j \rangle$ peut se déduire de deux expressions simples construites sur E_a, lorsque $b \leqslant a$.

Ce résultat sera utilisé dans la suite.

I.B – Manipulation de piles

En Caml, on définit le type `pile` (d'entiers) par :

```
type pile = {
  mutable elements: int list;
};;
```

En Pascal, le type `pile` est défini par :

```
type
  pmaillon = ^maillon;
  pile = pmaillon;

  maillon = record
    valeur: integer;
    suivant: pmaillon;
  end;
```

Le type `pile` possède trois opérations :
- `empile`, qui ajoute un élément au sommet de la pile ;
- `depile`, qui rend l'élément situé au sommet de la pile et le retire de la pile ;
- `vide`, qui rend vrai si et seulement si la pile est vide.

Écrire les fonctions `empile`, `depile` et `vide`.

I.C – Calcul des classes d'équivalence de E_1

On considère à partir de maintenant que A est composé des 26 lettres de l'alphabet. On suppose que l'on dispose d'une fonction `numero_lettre` qui à chaque lettre (caractère) associe son rang, compris entre 0 et 25.

On rappelle que si $i \in \{0, 1, \ldots, n-1\}$, la classe d'équivalence de i pour E_1 est le sous-ensemble X_i des entiers $j \in \{0, 1, \ldots, n-1\}$ tels que $\langle i, E_1, j\rangle$. Une classe d'équivalence de E_1 est une partie X de $\{0, 1, \ldots, n-1\}$ pour laquelle il existe $i \in \{0, 1, \ldots, n-1\}$ tel que $X = X_i$.

I.C.1) Pour $i, j \in \{0, 1, \ldots, n-1\}$, que signifie $\langle i, E_1, j\rangle$?

I.C.2) On note p le nombre de classes d'équivalence de E_1 ; les classes d'équivalence de E_1 seront numérotées par des entiers entre 0 et $p-1$. Elles seront représentées par un vecteur (Caml) ou tableau (Pascal) d'entiers, de longueur n (indices numérotés de 0 à $n-1$), dans lequel l'élément de rang i sera le numéro de la classe d'équivalence de i.

Dans la suite, on utilisera le terme vecteur pour désigner aussi bien les vecteurs de Caml que les tableaux de Pascal.

Exemple : pour le mot « abracadabra », le vecteur $[0, 1, 2, 0, 3, 0, 4, 0, 1, 2, 0]$ donne les classes d'équivalence de E_1.

Écrire une fonction `construit_E1` qui pour un mot m donné en argument, construit le vecteur représentant les classes d'équivalence de E_1 ; cette fonction utilisera un seul vecteur de longueur n et aucune autre structure annexe de type liste ou pile.

I.C.3) Donner la complexité de cette fonction en termes du nombre de lectures et d'écritures dans un vecteur.

I.C.4) On autorise maintenant la fonction `construit_E1` à utiliser un vecteur supplémentaire. Réécrire `construit_E1` de sorte qu'elle ait une complexité en $O(n)$.

I.D – Classes d'équivalence des E_{a+b}

On cherche dans cette section à construire les classes d'équivalence de E_{a+b}, connaissant les classes d'équivalence de E_a et sachant que $b \leqslant a$.

Comme précédemment, les p classes d'équivalence de E_a sont numérotées de 0 à $p-1$.

Soit v le vecteur donnant les numéros des classes d'équivalence de E_a ; on rappelle que v est un vecteur à n composantes. Le but de cette section est de construire w le vecteur à n composantes donnant les numéros des classes d'équivalence de E_{a+b}.

Pour construire les classes d'équivalence de E_{a+b}, on commence par regrouper les indices des lettres de m en sous-ensembles, de façon que deux éléments d, e quelconques d'un tel sous-ensemble vérifient $\langle d+b, E_a, e+b\rangle$. On remarque que seuls sont concernés les indices d vérifiant $d + b \leqslant n - a$.

I.D.1) Combien y aura-t-il de sous-ensembles au maximum ?

I.D.2) On souhaite construire ces différents sous-ensembles. On utilisera pour cela un vecteur de piles, chaque pile correspondant à un sous-ensemble. Expliquer comment on peut simplement construire ce vecteur de piles. *On ne demande pas d'écrire de programme en Caml ou Pascal à ce stade.* Certaines piles peuvent être vides.

I.D.3) Une pile donnée contient donc des indices tels que deux valeurs successives d et e vérifient $\langle d+b, E_a, e+b\rangle$. Si de plus ces deux valeurs successives vérifient $\langle d, E_a, e\rangle$, que peut-on en déduire d'utile pour construire les classes de E_{a+b} ?

I.D.4) Cependant, a priori la méthode de construction des piles de la question **I.D.2** ne garantit pas que tous les éléments d'une pile qui appartiennent à une même classe d'équivalence de E_a seront placés successivement. On cherche donc à modifier la solution de la question **I.D.2** pour ajouter cette propriété.

Nous allons commencer par créer un premier vecteur de piles (intermédiaire), contenant tous les éléments de v, et tel que si deux éléments d, e appartiennent à la même pile, alors $\langle d, E_a, e\rangle$. Écrire une fonction `empileClassesEa` qui fournit ce résultat, en fonction du vecteur v et du nombre p de classes d'équivalence de E_a.

I.D.5) À partir de ce vecteur de piles intermédiaire, modifier la méthode proposée à la question **I.D.2** de façon à obtenir un vecteur de piles final dans lequel les éléments d'une pile qui appartiennent à une même classe d'équivalence de E_a sont placés successivement. La solution sera réalisée sous forme d'une fonction `sousEnsembles` prenant en paramètre le vecteur v, vecteur de piles intermédiaire, et la valeur b.

I.D.6) En utilisant la propriété remarquée dans la question **I.D.3**, écrire une fonction `classesAplusB` qui calcule les classes d'équivalence de E_{a+b}. Cette fonction rendra le vecteur w et prendra en paramètre le vecteur v, le vecteur (final) de piles issu de la fonction de la question précédente et la valeur b.

I.D.7) Écrire finalement la fonction `classesAversAplusB` qui calcule le vecteur w à partir du vecteur v.

I.E – Construction des classes d'équivalence des E_k, $k > 1$

I.E.1) Soit q un entier naturel tel que 2^q soit inférieur ou égal à n. Montrer que l'on peut construire les classes d'équivalence de E_{2^q} en appliquant q fois `classesAversAplusB`.

I.E.2) Écrire une fonction `classesEq` qui, à un mot m de longueur n et à un entier quelconque $k \leqslant n$, associe les classes d'équivalence de E_k.

I.F – Sous-mots répétés

I.F.1) Expliquer comment `classesEq` permet d'obtenir la liste des sous-mots répétés d'une certaine longueur ℓ.

On cherche maintenant à trouver l'entier ℓ *maximum* pour lequel il y a des sous-mots répétés.

I.F.2) Quel test simple sur le résultat de `classesEq` permet de savoir si un mot m contient des sous-mots répétés de longueur ℓ ?

I.F.3) Nous avons vu en **I.E.1** que E_{2^q} peut être construit avec q appels à `classesAversAplusB`. En vertu de ce résultat et de la question précédente, on doit donc pouvoir déterminer le plus petit entier q tel qu'il n'existe pas de sous-mots répétés de longueur 2^q dans m.

Comment alors déterminer en un minimum d'opérations le plus grand entier ℓ ($\ell < 2^q$) tel que m contienne des sous-mots répétés de longueur ℓ ?

On ne demande pas de programmer cette opération.

II Langages et automates

II.A – Ensembles dénombrables

Notation — Pour tout ensemble A, on note $\mathcal{P}(A)$ l'ensemble des sous-ensembles de A. En particulier, $A \in \mathcal{P}(A)$.

Définitions — Deux ensembles A et B sont dits équipotents, s'il existe une bijection de A vers B. Un ensemble A est dit dénombrable si A et $\mathbb{N}$ sont équipotents ($\mathbb{N}$ désigne l'ensemble des entiers naturels).

II.A.1) Si A est un ensemble fini, montrer que A et $\mathcal{P}(A)$ ne sont pas équipotents.

II.A.2) Si A est un ensemble quelconque, montrer que A et $\mathcal{P}(A)$ ne sont pas équipotents.

Indication. On peut raisonner par l'absurde en supposant qu'il existe une bijection $f : A \to \mathcal{P}(A)$; on considérera $B = \{a \in A \mid a \notin f(a)\}$ et b tel que $f(b) = B$, puis on examinera la véracité de $b \in B$.

II.A.3) Soit $\Sigma = \{0\}$.

Montrer que $\mathcal{P}(\Sigma^*)$ n'est pas dénombrable.

II.A.4) On peut montrer que l'ensemble des langages rationnels est dénombrable. Cela implique qu'il existe des langages sur $\{0\}$ qui ne sont pas rationnels. En voici un :

Soit $L_P = \{0^i \mid i \text{ est premier}\}$. Montrer que L_P n'est pas rationnel.

II.B – Lemme d'Arden

Le lemme d'Arden permet de résoudre des équations du type : $X = (A \cdot X) \cup B$ où A, B et X sont des langages.

Soit Σ un alphabet.

Définition Soit A, $B \subset \Sigma^*$ deux langages sur Σ. Le langage $A \cdot B$ est défini par

$$A \cdot B = \{u \in \Sigma^* \mid \exists v \in A, \exists w \in B \ / \ u = vw\}$$

Le langage A^* est défini comme l'union $\bigcup_{i \in \mathbb{N}} A^i$ avec :

$$\begin{aligned} A^0 &= \{\varepsilon\} \\ A^{i+1} &= A \cdot A^i \end{aligned}$$

II.B.1) Soit I un ensemble non vide et, pour chaque $i \in I$, soit A_i un langage sur Σ. Soit B un langage sur Σ. Montrer les propositions suivantes :

$$\left(\bigcup_{i \in I} A_i\right) \cdot B = \bigcup_{i \in I} (A_i \cdot B) \tag{II.1}$$

$$B \cdot \left(\bigcup_{i \in I} A_i\right) = \bigcup_{i \in I} (B \cdot A_i) \tag{II.2}$$

II.B.2) On désigne par (E) l'équation

$$X = (A \cdot X) \cup B$$

où $A \subset \Sigma^*$ et $B \subset \Sigma^*$.

Démontrer que $A^* \cdot B$ est une solution de (E).

II.B.3) Démontrer que, pour l'inclusion, $A^* \cdot B$ est la plus petite solution de (E).

II.B.4) Donner un exemple de valeurs de A et B pour lesquelles l'équation (E) admet plusieurs solutions.

II.B.5) Démontrer que si $\varepsilon \notin A$ alors $A^* \cdot B$ est l'unique solution de (E).

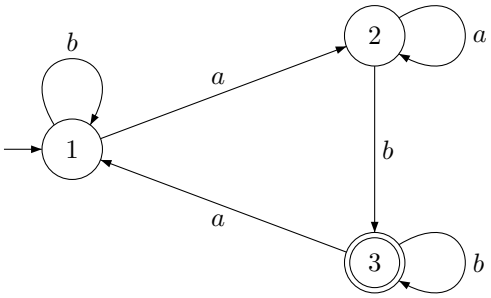
II.B.6) Un automate fini déterministe $\mathcal{A}$ est décrit par une structure $(Q, \Sigma, \delta, q_I, F)$ avec :
- Q l'ensemble fini non-vide des états de $\mathcal{A}$;
- Σ l'alphabet ;
- $\delta : Q \times \Sigma \to Q$ la fonction de transition ;
- q_I l'état initial ;
- $F \subset Q$ l'ensemble des états terminaux.

À chaque état $q \in Q$ on associe l'équation $\mathrm{E}(q)$:

$$\begin{cases} X_q = \bigcup_{a \in \Sigma} \left(\{a\} \cdot X_{\delta(q,a)} \right) & \text{si } q \notin F \text{ et} \\ X_q = \left(\bigcup_{a \in \Sigma} \left(\{a\} \cdot X_{\delta(q,a)} \right) \right) \cup \{\varepsilon\} & \text{si } q \in F. \end{cases}$$

Le système d'équations $\mathrm{SE}(\mathcal{A})$ associé à $\mathcal{A}$ est l'ensemble qui contient les équations $\mathrm{E}(q)$ pour $q \in Q$.

On considère, et on représente ci-dessous, l'automate $\mathcal{A} = (\{q_1, q_2, q_3\}, \{a, b\}, \delta, q_1, \{q_3\})$ avec $\delta(q_1, a) = q_2$, $\delta(q_1, b) = q_1$, $\delta(q_2, a) = q_2$, $\delta(q_2, b) = q_3$, $\delta(q_3, a) = q_1$ et $\delta(q_3, b) = q_3$.



Écrire le système d'équations associé à $\mathcal{A}$.

II.B.7) Calculer une solution de $\mathrm{SE}(\mathcal{A})$.

II.B.8) Que représente le langage associé à q_1 par la solution de $\mathrm{SE}(\mathcal{A})$?

• • • FIN • • •