Algorithmique III (I41) - Licence d'Informatique

Contrôle terminal (Session 1 - Mai 2018)

Soit $\mathscr{A}$ un alphabet fini et $n \in \mathbb{N}$. On désigne par $\mathscr{A}^*$ l'ensemble des mots définis sur l'alphabet $\mathscr{A}$ et par $\mathscr{A}^n$ l'ensemble des mots de longueurs n. On rappelle que $\mathfrak{S}_n$ désigne le groupe des permutations de degré n, c'est-à-dire l'ensemble des applications bijectives de $[1, n]$ dans lui-même muni de la loi de composition des applications.

Soit u et v deux mots de $\mathscr{A}^*$. On dit que v est une *anagramme* de u si et seulement si u et v ont même longueur $n = |u| = |v|$ et s'il existe une permutation $\sigma \in \mathfrak{S}_n$ telle que

$$\forall i \in [1, n] \quad v_i = u_{\sigma(i)}. \tag{1}$$

Ce que l'on résume par $v = \sigma(u)$.

(1) Combien d'anagrammes distinctes existe-t-il d'un mot de longueur n dont toutes les lettres sont distinctes ? Même question, mais $n \geqslant 3$ et le mot contient exactement trois lettres identiques ?

(2) Écrivez un algorithme `TriAlphaMot`(mot) de complexité *linéaire*, qui renvoie le mot passé en paramètre trié dans l'ordre alphabétique. Par exemple `TriAlphaMot`("*sacres*") renvoie le mot "*acerss*".

Indications : vous supposerez que $\mathscr{A} = \{a, b, c, \ldots, z\}$ et vous utiliserez, *sans l'écrire*, un algorithme `Ordre`($lettre$) qui renvoie la position de `lettre` dans l'alphabet, ainsi `Ordre`("a") renvoie 0 et `Ordre`("z") renvoie 25. Vous noterez $+$ la concaténation des mots et $\leqslant$ l'ordre alphabétique sur $\mathscr{A}$.

(3) En déduire un algorithme `EstAnagramme`(u, v) qui décide (i.e. renvoie *vrai* ou *faux*) si le mot v est une *anagramme* du mot u. Quelle est sa complexité ?

(4) Écrivez un algorithme `AlphaPerm`(u) de complexité *linéaire* qui renvoie une permutation $\sigma \in \mathfrak{S}_n$ qui satisfait (1) où $v =$ `TriAlphaMot`(u).

(5) En déduire un algorithme `ChercherPerm`(u, v) qui renvoie une permutation $\sigma \in \mathfrak{S}_n$ qui satisfait (1) en supposant que v est une anagramme de u et que $n = |u| = |v|$. Quelle est sa complexité ?

(6) Démontrez que la relation binaire $\bowtie$ définie sur $\mathscr{A}^*$ par

$$u \bowtie v \;\Leftrightarrow\; \exists n \in \mathbb{N} \;\; \exists \sigma \in \mathfrak{S}_n \quad (n = |u| = |v|) \;\wedge\; (v = \sigma(u)).$$

est une *relation d'équivalence*. Quelles sont les classes d'équivalence du sous-ensemble

$$M := \{\text{"}se\text{"}, \text{"}car\text{"}, \text{"}test\text{"}, \text{"}mais\text{"}, \text{"}arc\text{"}, \text{"}es\text{"}, \text{"}amis\text{"}\}$$

de $\mathscr{A}^*$? Plus généralement, que représentent les classes d'équivalence pour cette relation ?

(7) On se donne un sous-ensemble fini M de mots de $\mathscr{A}^*$ codé par une liste. Écrivez un algorithme `Partitionner`(M) qui renvoie l'ensemble $M/\bowtie$ des classes d'équivalence de M pour la relation $\bowtie$.

(8) Calculez la complexité de l'algorithme `Partitionner`.