

	ANNÉE UNIVERSITAIRE 2007–2008			
	SESSION DE JUIN 2008			
	<b>Parcours</b>	MATMECA 1ère année	<b>Code UE</b>	MMK2I1
	<b>Date</b>	2 juin 2008, 14h00	<b>Durée</b>	1h30
	<b>Documents</b>	Transparents, notes de cours/TD autorisés.		
<b>Épreuve</b>	Algorithmique et programmation. M. Zeitoun			

Les programmes sont à écrire en pseudo-langage vu en cours (pas en fortran). La rigueur et la clarté des justifications ainsi que la qualité de la présentation seront des éléments d'appréciation. Par complexité des algorithmes, on entend complexité dans le cas le pire.

1. En utilisant le type abstrait *liste* vu en cours, écrire une fonction prenant en argument deux listes  $L_1$  et  $L_2$ , et retournant leur concaténation, c'est-à-dire la liste contenant les éléments de  $L_1$  suivis par ceux de  $L_2$ . Par exemple, si  $L_1 = (2, 1, 0)$  et  $L_2 = (5, 1)$ , la fonction retournera la liste  $(2, 1, 0, 5, 1)$ . On demande que la complexité de la fonction écrite ne dépende *pas* de  $L_2$ .
2. Évaluer la complexité de la fonction précédente en termes du nombre d'appels à une opération du type abstrait. Justifier.
3. Décrire une structure de donnée permettant de représenter le type abstrait *liste*, tout en obtenant une complexité  $O(1)$  pour la concaténation de deux listes. Expliquer précisément la façon dont la concaténation est réalisée (on ne demande cependant pas d'écrire du code).

Soit  $A = \{k \in \mathbb{N} \mid k < K\} = \{0, 1, \dots, K - 1\}$ , où  $K$  est un entier positif. On note  $A^+$  l'ensemble des suites finies non vides d'éléments de  $A$ . Par exemple, si  $K = 3$ , on a  $(2, 0, 0, 1) \in A^+$ . Un élément de  $A^+$  est appelé un *mot*. Pour  $u \in A^+$ , on note  $|u|$  la longueur de la suite  $u$ . Par exemple,  $|(2, 0, 0, 1)| = 4$ . La  $i$ -ème lettre de  $u$  est notée  $u[i]$ , en commençant la numérotation à 0, de sorte que  $u = (u[0], u[1], \dots, u[|u| - 1])$ . Cette notation est justifiée par le fait que les mots seront stockés dans des tableaux.

L'ordre *lexicographique* sur  $A^+$  est défini ainsi : pour  $u, v \in A^+$ , on a  $u <_{\text{lex}} v$  si et seulement si

- soit il existe  $k < \min(|u|, |v|)$  tel que  $u[k] < v[k]$ , et pour tout  $i$  tel que  $0 \leq i < k$ , on a  $u[i] = v[i]$  ;
- soit  $|u| < |v|$  et pour tout  $i$  tel que  $0 \leq i < |u|$ , on a  $u[i] = v[i]$ .

4. Classer dans l'ordre lexicographique  $<_{\text{lex}}$  les mots suivants :  $u = (0, 1, 6, 4)$ ,  $v = (0, 1, 4)$ ,  $w = (0, 3)$ ,  $x = (2)$ ,  $y = (0, 1)$ .

On se propose de trier efficacement une suite de mots  $X = (x_1, x_2, \dots, x_n)$  dans l'ordre lexicographique. On suppose que cette suite est donnée par une liste dont chaque cellule contient une structure décrivant un mot. Une telle structure a deux champs : le mot, stocké dans un tableau, et sa longueur. La première cellule contient le mot  $x_1$  et sa longueur, la seconde le mot  $x_2$  et sa longueur, etc.

On suppose d'abord que chaque mot  $x_i$  est de longueur 1. Ainsi,  $x_i[0]$  représente l'unique lettre de  $x_i$ . On se propose d'utiliser l'algorithme suivant pour trier les mots :

- a. On crée un tableau  $T$  de  $K$  cases  $T[0], T[1], \dots, T[K-1]$ .
  - b. On initialise chaque case avec une liste vide.
  - c. On parcourt la liste  $X$  des mots (de longueur 1) à trier, et pour chacun de ses éléments  $x_i$ , on remplace  $T[x_i[0]]$  par la concaténation de  $(x_i)$  et  $T[x_i[0]]$ .
  - d. On retourne la concaténation des listes  $T[0], T[1], \dots, T[K-1]$ , dans l'ordre.
- 5.** Simuler l'exécution de l'algorithme sur la liste de mots (chacun de longueur 1) suivante :  $((7), (4), (7), (1), (0), (4), (7))$ , pour  $K = 10$ . Décrire en particulier l'état de  $T$  avant l'étape  $d$ .
- 6.** Écrire formellement l'algorithme en pseudo-langage. Quelle est sa complexité dans le cas le pire ?
- 7.** Adapter l'algorithme au cas où tous les mots ont la même longueur  $\ell$ , supposée connue. On ne demande pas d'écrire l'algorithme mais de l'expliquer en français. On demande que la complexité soit  $O(n\ell + K\ell)$ , où  $n$  est la longueur de la liste  $X$  à trier.
- 8.** Justifier la borne inférieure suivante pour ce problème : tout algorithme de tri dans l'ordre  $<_{\text{lex}}$  d'une liste  $X = (x_1, x_2, \dots, x_n)$  de  $n$  mots a pour complexité au moins  $\Omega\left(\sum_{i=1}^n |x_i|\right)$ .
- 9.** Comment adapter l'algorithme dans le cas où les mots n'ont pas tous la même longueur ?
- 10.** (Difficile) Montrer qu'on peut trier une liste de mots, tous de longueur  $\ell$ , en temps  $O(n\ell + K)$ .

*Indication.* Une possibilité est de calculer, pour chaque position  $j$  entre 1 et  $\ell$ , l'ensemble trié des valeurs apparaissant dans l'une des suites de  $X$ , à position  $j$ . On pourra pour cela associer à chaque mot  $u = (u[0], \dots, u[\ell-1])$  de  $X$  la suite de couples d'entiers  $\tilde{u} = ((0, u[0]), \dots, (\ell-1, u[\ell-1]))$ , et trier dans l'ordre lexicographique tous les tels couples apparaissant dans les suites  $\tilde{x}_i$ .