Université Denis-Diderot
Licence I
Informatique: IF2

Examen Informatique IF2

Première session, durée 3 heures.
Tous les documents sont interdits.
(2+8 pages)

Cette examen comporte un questionnaire à choix multiples sur 8 pages dont vous rendrez la première page avec votre copie.

Les questions sont indépendantes.

1. Un mot u est une suite de lettre $a_0a_1 \dots a_n$ (chaque a_i est un caractère). On définit $rev(u)$ le mot u lu de droite à gauche: pour $u = a_0a_1 \dots a_n$ alors $rev(u) = a_na_{n-1} \dots a_0$.

   On représentera un mot par une `String` java. On rappelle que la classe `String` contient (entre autres) la méthode `charAt(int i)` qui retourne le caractère qui se trouve en i-ème position dans la chaîne, la méthode `substring(int debut, int fin)` qui retourne la sous-chaîne commençant en position `debut` et se terminant en position `fin-1`, (par exemple `"abcd".susbtring(1,2)="bc"`). Rappelons aussi que `length` retourne le nombre de caractères de la chaîne.

   Exemples: on a `"abcd".susbtring(1,2)="bc"`. Et si `st="eluparcettecrapule"` alors `st.charAt(2), st.length(), st.substring(2,12)` valent respectivement u, 18 et "uparcettec".

   On rappelle aussi que l'opérateur `+` appliqué à des objets `String` réalise une "concaténation": si $u = a_0a_1 \dots a_n$ et $v = a'_0a'_1 \dots a'_m$ alors $u + v = a_0a_1 \dots a_na'_0a'_1 \dots a'_m$.

   D'autre part, un mot est un *palindrome* s'il se lit de la même manière de droite à gauche et de gauche à droite. Par exemple "eluparcettecrapule" est un palindrome.

   (a) Écrire une méthode itérative prenant en entrée un mot u quelconque qui retourne $rev(u)$.

   (b) Écrire une méthode récursive prenant en entrée un mot u quelconque qui retourne $rev(u)$.

   (c) Écrire une méthode itérative permettant de tester si un mot est un palindrome.

   (d) Remarquons qu'un mot de moins d'une lettre est un palindrome et qu'un mot est un palindrome si (1) sa première et sa dernière lettre sont identiques et (2) le mot privé de sa première lettre et de sa dernière lettre est un palindrome. En déduire une méthode récursive permettant de tester si un mot est un palindrome.

2. Le but de cette question est de trier de façon très rapide un tableau d'entiers dans le cas particulier où l'ensemble des valeurs est connu et est de taille raisonnable. On suppose donc ici que les valeurs à trier sont des entiers positifs ou nuls inférieurs à une valeur fixe n. On dispose donc d'un tableau t d'entiers tel que pour tout indice i du tableau on a: $0 \leq t[i] < n$. Le principe de ce tri est très simple: on remplit un tableau auxiliaire *temp* de n éléments qui compte le nombre d'occurrences de chaque valeur. Quand le tableau est rempli on aura $temp[i] = j$ s'il y a exactement j éléments égaux à i dans t. On peut ensuite en parcourant *temp* obtenir le tableau résultant du tri de t.

   (a) Écrire une méthode statique *tri* ayant comme arguments un tableau d'entiers t et un entier n qui trie ce tableau avec l'algorithmique indiqué ci-dessus si tous les éléments de ce tableau sont des entiers positifs ou nuls inférieurs à n.

   (b) Donner une pré-condition et une post-condition correspondant à la méthode de la question précédente.

(c) Si on suppose que tous les éléments de t sont distincts, quelle est (en fonction de la taille du tableau t et de n) l'ordre de grandeur du nombre d'opérations réalisées (pour le pire cas, le meilleur cas et pour le cas moyen)? Quel est pour votre algorithme l'ordre de grandeur du nombre d'opérations réalisées dans le pire cas si les éléments de t ne sont pas tous distincts?

(d) On suppose maintenant qu'au lieu d'un tableau on dispose d'une liste chaînée d'entiers comme définie ci-dessous:

```
class Cellule {
    public Cellule suivant;
    public int val;
    public Cellule(int i, Cellule c) {
        val = i;
        suivant = c;
    }
    public Cellule(int i) {
        this(i, null);
    }
}
```

Ré-écrire dans ce cas l'algorithme précédent. On écrira de plus une méthode permettant d'afficher le contenu de la liste.