IN 103 - TD 01

Énoncé

Pour ce TD, il vous est donné le fichier `french.txt` qui contient une liste de 239185 mots en Français triés par ordre alphabétique.

Le but de ce TD est d'appliquer les techniques de recherche en table vue en cours magistral. Vous devrez constater la différence d'efficacité (donc de complexité) entre deux méthodes de recherche.

Dans le fichier `load_dic.(c|h)` vous est donnée la fonction :
char ∗∗load_file (**char** ∗fname, **unsigned int** ∗size)
qui permet de charger le contenu d'un fichier de mots en mémoire dans un tableau.

Q1 À partir du prototype donné de la fonction `load_file`, déduisez ce que fait cette fonction en terme de structures de données. Pourquoi ces types de paramètres et de retour ?

Q2 La comparaison de chaînes de caractères en C ne se fait pas avec l'opérateur `==`. Pourquoi à votre avis ?

Pour comparer 2 chaînes de caractères, on utilise la fonction :
int strcmp (**char** ∗s1, **char** ∗s2)
qui retourne 0 si les chaînes sont égales, une valeur < 0 si `s1` $<$ `s2`, une valeur > 0 si `s1` $>$ `s2`.

1 Recherche naïve

Q3 Écrivez une fonction qui prend en argument un mot et le tableau de mots et retourne un booléen disant si le mot a été trouvé dans le tableau. Cette fonction fera une recherche séquentielle dans le tableau.

Q3 Quelle est la complexité de cette fonction en nombre de comparaisons ?

Dans la suite, on voudra lire des mots au clavier jusqu'à ce que ^ -d ait été pressé. La fonction :
char∗ gets (**char** ∗str)
permet de lire une chaîne au clavier en la rangeant dans le tableau (pré-alloué par vous) `str` et retourne un pointeur vers ce tableau. Le pointeur retourné permet juste de déterminer si une chaîne a bien pu être lue : si oui, la valeur retournée est $\neq$ `NULL`, sinon, la valeur retournée est $=$ `NULL` et cela signifie que la ≪fin de fichier≫ et atteinte (pression de ^ -d au terminal).

Q4 Complétez votre programme avec un `main ()` qui permette de saisir des mots et de vérifier s'ils existent dans le dictionnaire. Pour arrêter le programme, il suffira de presser ^ -d au terminal.

Q5 Testez votre programme. Il se peut qu'en fonction du codage des lettres accentuées, les mots contenant des accents ne soient pas trouvés.

Q6 Maintenant, petit test de performance, testez que tous les mots du dictionnaire sont bien dans le dictionnaire. Comment allez-vous vous y prendre ?

2 Recherche dichotomique

Dans cette section, nous allons implémenter une recherche dichotomique dans le tableau.

Q7 Esquissez l'algorithme. Quel sont les cas d'arrêt, quelle est la structure des traitements ?

Q8 Implémentez la fonction de recherche par dichotomie.

Q9 Refaites le test de la question **Q6**.

Q10 Quelle est la complexité de cette fonction en nombre de comparaisons ?