Assistance de la saisie de messages sur les téléphones portables :

Rapport

Présentation

Le projet porte principalement sur l'implémentation de deux arbres : l'un permettant d'effectuer des recherches à travers un dictionnaire, l'autre étant utilisé pour la compression et la décompression d'un fichier au moyen de l'algorithme de Huffman. Le présent projet est appelé « Act », pour *Auto-Completion Tree*, en référence à l'arbre gérant le dictionnaire pour l'auto-complétion en cours de frappe d'un texte.

Act se présente sous la forme d'une fenêtre comportant des menus, une zone de texte, une liste et un bouton *Insérer*. C'est dans la boîte d'édition de texte que l'on tape les mots qui pourront être complétés. En effet, au fur et à mesure de la frappe du texte, s'affichent dans la liste (à droite de la fenêtre) les dix meilleures propositions de complétion pour le mot en cours. Si l'une d'entre-elles correspond et que l'on souhaite l'insérer dans le texte, il suffit de la sélectionner puis de cliquer sur le bouton *Insérer*.

Il est intéressant de noter que chaque mot complété, c'est-à-dire dès que l'on entre autre chose qu'un caractère alphabétique après une suite de tels caractères, est ajouté automatiquement au dictionnaire. Si ce mot y est déjà présent, son score dans le dictionnaire sera augmenté et il apparaîtra plus haut dans la liste des propositions. De même, lorsqu'un fichier est ouvert, chaque mot qui le compose est ajouté dans le dictionnaire de la même manière que s'ils avaient été tapés.

Les menus servent à effectuer diverses opérations sur les fichiers, le texte et le dictionnaire, à savoir :

- menu *Fichier* : effacer, ouvrir et enregistrer le texte ; quitter le programme ;
- menu *Édition* : couper, copier et coller du texte ; sélectionner tout le texte ;
- menu *Dictionnaire* : effacer, ouvrir et enregistrer le dictionnaire ; ajouter des mots issus d'un autre dictionnaire au dictionnaire courant ;
- menu *Aide* : afficher quelques informations sur Act.

Le dictionnaire

Le dictionnaire est implémenté sous la forme d'un arbre de recherche ternaire. Ce type d'arbre n'est pas beaucoup plus compliqué à implémenter qu'un arbre binaire, mais son principal avantage est de permettre, dans le meilleur des cas, d'effectuer une recherche dichotomique sur un caractère, et, dans le pire des cas, de le faire linéairement. Autrement dit, par rapport à l'arbre binaire, l'arbre ternaire ne peut qu'être plus performant ou aussi performant. La configuration où l'arbre ternaire effectue les recherches de façon linéaire se produit lorsque les mots sont entrés dans le dictionnaire en respectant l'ordre alphabétique (ou l'ordre alphabétique inversé).

Dans un arbre de recherche binaire, chaque nœud pointe vers un frère et un fils. Le frère correspond en fait à un autre caractère à la même position dans la chaîne de caractères recherchée et le fils est le caractère suivant dans la chaîne. Dans un arbre ternaire, par contre, il y a deux frères et un fils. La raison pour laquelle il y a deux frères permet justement de faire une recherche dichotomique entre les frères : le premier frère est toujours inférieur au nœud courant tandis que le second lui est toujours supérieur. C'est le même principe qui est utilisé dans le tri rapide (*quicksort*).

Une étude sur les arbres de recherche ternaires peut être trouvée sur ce site : `http://www.cs.princeton.edu/~rs/strings/paper.pdf`.

La recherche des meilleures propositions dans le dictionnaire s'effectue par le biais d'une fonction « callback ». On utilise une fonction qui va parcourir tout l'arbre et appeler la fonction « callback » à chaque fois qu'un mot est trouvé. Cette méthode est également utilisée pour convertir un dictionnaire en une chaîne de caractères (en vue de le sauvegarder dans un fichier par exemple).

La compression/décompression

L'algorithme de compression de Huffman se base sur un codage plus court des caractères fréquents et plus long des caractères rares. C'est une méthode simple et efficace, très répandue. Pour ce faire, un arbre binaire est utilisé. À chaque caractère correspond un code, qui est construit en parcourant l'arbre de la racine au noeud. Lors de la compression, les codes sont créés en empruntant le chemin inverse.

Pour la compression, une queue (ou file) de priorité a été implémentée. Celle-ci permet de rajouter et de retirer très rapidement des éléments à une liste dont le

premier est toujours le plus petit (ou le plus grand, suivant l'utilisation de cette queue). Ces opérations se font avec une complexité de l'ordre de $O(\log_2(n))$, ce qui est bien plus optimal qu'avec une méthode par insertion/décalage d'une complexité de $O(n)\ldots$

La compression se fait en créant d'abord un arbre composé de feuilles correspondant aux caractères présents dans le texte à compresser. L'algorithme de création de l'arbre complet est ensuite exécuté. Puis, pour chaque caractère, l'arbre est parcouru à partir des feuilles jusqu'à la racine afin de créer une table contenant le code de chaque caractère : cela permet d'écrire le fichier plus rapidement qu'en reparcourant l'arbre pour chaque caractère. Par contre, pour la décompression, l'arbre est parcouru pour chaque caractère.

Le format du fichier compressé est le suivant :
- en-tête « HUFF » de 4 octets ;
- la taille du fichier décompressé, un entier 32-bit non signé stocké en *little endian* ;
- le dictionnaire des codes tirés de l'arbre de HUFFMAN : voir ce-dessous ;
- les données compressées.

Le dictionnaire est stocké sous la forme d'un tableau de bits. Pour chaque caractère, allant des codes ASCII 0 à 255, sont présents 8 bits indiquant la longueur du code ainsi que le code lui-même. Ces données ne sont pas alignées, mêmes pas lors du passage de la table des codes aux données compressées. Tout est collé bit à bit ; ceci permet un certain gain de place en ne laissant aucun bit inutilisé.

Les fichiers générés par ces fonctions sont en moyenne deux fois plus petits que les originaux. De plus, un fichier pourra être lu sur toutes les architectures puisque les algorithmes ont été conçus pour ne pas souffrir d'incompatibilités dues aux différences d'endianisme entre les machines.