UE 603 Complexité et Algorithmique
L3 SIO 2011-12

Projet : recensement des mots d'un corpus, analyse de structures de données

Date limite : 16 avril 2012

PRESENTATION DU PROBLEME

De nombreuses applications du Traitement Automatique du Langage, mais aussi des applications maintenant quotidiennes (comme le mode T9 d'un téléphone mobile) demandent un accès efficace à un lexique.

Dans ce projet, vous allez étudier les performances théoriques et pratiques de plusieurs algorithmes et structures de données dans l'objectif de recenser les mots d'un corpus (ensemble de fichiers) textuel ainsi que la fréquence de chacun.

LES DONNEES

Nous allons utiliser les données de wikipédia.

Ce site fournit régulièrement une sortie de l'ensemble de sa base (voir http://dumps.wikimedia.org/frwiki/20111103/). Ce « dump » est dans un format un peu compliqué. Vous allez travailler sur un format plus simple : suppression des balises XML, conversion du données wikimédia en texte simple. Notez bien que le résultat obtenu n'est pas parfait (beaucoup de modèles wikimédia sont perdus, etc), mais ce n'est pas important pour la tâche qui vous incombe.

Vous trouverez le fichier des données de wikipédia à l'adresse : https://transfert.inria.fr/fichiers/ace461433076c6aa43cabe3d21fdb65f/wikipediadump.gz. Attention, ce fichier contient 225 millions de mots et a une taille de 455Mo zippé.

TRAVAIL A FAIRE

Votre objectif est de lister les mots présents dans le corpus avec l'occurrence de chacun.

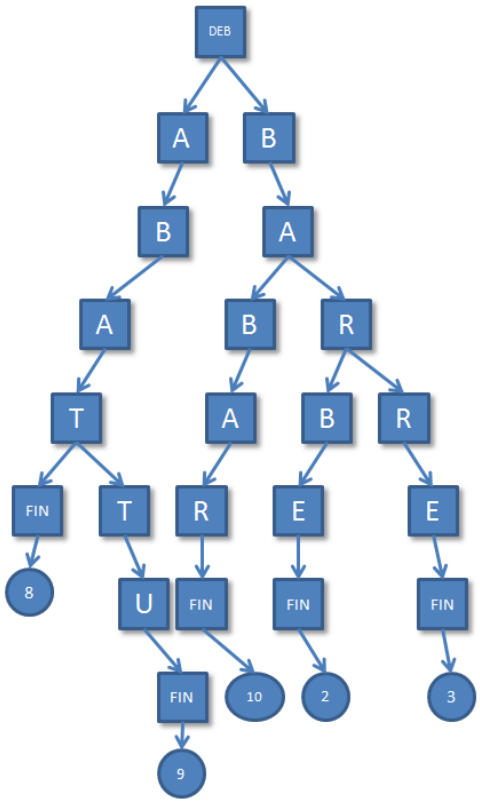
Pour ce travail, nous appelerons « mot » toute suite de caractères précédée et suivie des caractères ' ' (espace), '\n' (retour à la ligne), '\t' (tabulation), ou EOF (fin de fichier).

L'algorithme en lui-même est très simple. Le problème est de trouver une structure de données efficace en accès (un mot a-t-il déjà été rencontré ?) et en insertion (insertion d'un mot rencontré pour la première fois). Vous allez donc devoir comparer les performances en terme de temps de calcul et de place mémoire de plusieurs structures :

- **Table de hachage**, avec la fonction de hachage standard de Java, ainsi qu'une autre que vous pourrez développer en tenant compte des spécificités du problème (la répartition des fréquences des mots d'une langue n'est pas uniforme, ou encore la répartition des longueurs des mots n'est pas uniforme, ou encore,... toute idée novatrice que vous pourriez trouver).
- **Arbre Binaire non équilibré**
- **AVL** : la programmation d'une classe AVL n'est pas évidente. Vous pouvez utiliser une classe trouvée sur internet à condition que la ressource soit libre (type creative

commons). Vous pouvez aussi adapter l'implémentation en C fournie en pièce jointe sous le sujet.

- **Une structure originale** : le lexique est stocké dans un arbre. Une branche de l'arbre correspond à un mot. Un noeud à une lettre. Un noueud peut avoir plusieurs fils. Le chemin partant du sommet à un noeud correspond à un préfixe de mots, et les sous-arbres fils de ce noeud rassemblent tous les mots ayant ce préfixe. Par exemple le lexique : « abat » 8 fois, « abattu » 9 fois, « barbe » 2 fois, « barre » 3 fois, « babar », 10 fois, serait représenté comme suit :



Vous devrez donc développer cette structure de données. Les détails de développement devront être choisis de manière à maximiser les performances (notamment la structure interne d'un noeud).

QUESTIONS

Vous devrez répondre aux questions suivantes :

- Pour chaque structure :
  - Quels sont les paramètres qui vont avoir un impact sur les performances (en temps de calcul, en coût en mémoire).
  - Quelle est la complexité en O de l'insertion et de la recherche ? Même si vous n'arrivez pas à donner une réponse précise, donnez des éléments de réflexion qui nous montrerons que vous avez une idée assez précise des forces et faiblesses de chaque structure.
  - Quel est le temps de calcul réel ? (Pensez à faire plusieurs tests et à moyenner de manière à diminuer les influences extérieures de la gestion du système d'exploitation, du fonctionnement du ramasse miettes). Quel est le coût en mémoire ?

- Montrez à l'aide d'un graphique le coût en mémoire de la structure de données en fonction de la taille du fichier (on fera varier la taille de 1 à N où N est le nombre de mots du fichier). Proposer une explication à la courbe que vous obtenez.
- Montrez à l'aide d'un graphique le coût d'appel à la fonction d'insertion en fonction de la taille de l'entrée ; quel est le bon paramètre : le nombre de mots N dans le fichier, ou la taille de la structure de donnée ? En fonction de quoi peut-on exprimer cette taille ? Proposez une explication à la courbe que vous obtenez.

- Pour les ABR, quelle est la hauteur effective de l'arbre ? Cela correspond à quel facteur multiplicatif par rapport à la hauteur d'un AVL ? Le déséquilibre induit a-t-il un impact fort sur les performances par rapport à une AVL ? Pourquoi ?

POINT IMPORTANTS

Vous devez remettre un programme Java documenté (avec commentaires clairs et courts) ainsi qu'un document. Il est important que votre programme suive les principes de la programmation orientée objet.

Votre document doit être clair et structuré (faites attention à la présentation). Il doit contenir :

- les renseignements nécessaires pour une bonne utilisation du programme
- **une description des objets (structures, classes) et des algorithmes**
- un dossier de tests : les tests qui ont été effectués pour garantir le bon fonctionnement du programme (discuter entre les groupes des valeurs que vous devriez obtenir, et convergez).
- les réponses à toutes les questions posées de façon à ce que le lecteur trouve facilement les réponses dans votre document.
- **Vous êtes 4/5 par groupe : il y a moyen de vous répartir facilement le travail (segmentation en structure de données). Utilisez cette possibilité.**

Vous devez travailler en groupes de 5 (exceptionnellement 4) et valider votre groupe par email au plus tôt. Votre projet doit être remis pour le 16 avril 2012 à minuit (à déposer dans l'interface de l'ENT). Les soutenances auront lieu une semaine plus tard.