POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S6) / 2013 **Programmation Avancée**

Jonathan Aceituno, Vincent Aranéga, Nathalie Devésa, Rachid Elkouri et Laure Gonnord

TP9 Arbres d'entiers

Objectifs

- Savoir utiliser une bibliothèque d'arbres d'entiers.
- Savoir coder quelques algorithmes de base sur les arbres.

Contexte et préparation Dans ce TP, nous allons récupérer une bibliothèque d'arbres fournie, et la compléter. Nous utiliserons ensuite les fonctions écrites pour générer des dessins d'arbres en pdf. TP proposé par B. Carré.

Préparation

Avant de commencer, copier le répertoire `~lgonnord/PA2012/TP11/`. Vous devez récupérer :

- la bibliothèque `trees.c` et son fichier d'interface `trees.h`.
- un fichier `main.c` utilisant ces fonctions.
- un Makefile permettant de compiler ces fichiers.
- un sous-répertoire `samples` qui contient: des fichiers de données : `unspecified.txt, balanced.txt, degenerated.txt, empty.txt, leaf.txt` et des fichiers `.dot` et `.pdf` qui permettent de visualiser les arbres correspondant à ces données (section 2)
- un sous-répertoire `part2` contenant un autre `main.c` et un Makefile.

1 Fonctions de base dans les ABR

Dans cette première partie, il vous est demandé d'implémenter les fonctions suivantes non implémentées dans `trees.h`, en utilisant le main fourni pour tester :

1. Ajout d'un élément dans un ABR (arbre binaire de recherche, donc *trié*).
2. Construction d'un ABR à partir d'un fichier de données. Pour cette fonction, on s'inspirera de la fonction `chargeFichier` donnée dans le TP8 (attention on ajoute des entiers et non des chaînes).
3. Impression d'un ABR par parcours récursif (uniquement les noeuds).
4. Vérifier que la fonction de parcours précédente donne la même suite de valeurs pour les trois fichiers `unspecified.txt, balanced.txt, degenerated.txt` :
   `1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15`
   Alors que ce sont des arbres différents (voir les dessins dans le répertoire `samples`)
5. Écrire une autre fonction d'impression qui imprime les couples $(pere, fils)$ d'un arbre donné. On fera attention à ne pas imprimer les sous-arbres vides.
6. Vérifier que l'impression donne des arbres correspondant aux fichiers pdf fournis dans le répertoire `samples`.

2 Impression des arbres sous forme graphique

Les fichiers `.pdf` ont été générés à partir des fichiers `.dot` (fournis dans `samples/`). Le format `.dot` est un format de représentation textuelle simple d'arbres[1], voir le format de ces fichiers dans votre éditeur de texte préféré ou par la comande `more`[2]. A partir de ce format `.dot`, la commande `dot` permet de gérérer une version visualisable (`.pdf`, `.png`, ...) comme suit (`en pdf` par exemple)`bash> dot -Tpdf samples/balanced.dot -o samples/balanced.pdf`

L'objectif est alors de générer de tels fichiers pour les arbres construits à partir de fichiers de données (tels que ceux fournis dans `samples/*.txt`).

Préparation Copier votre bibliothèque de la première partie (`trees.c` et `trees.h`) dans le répertoire `part2`. Dans ce répertoire, un `main.c` plus conséquent vous est fourni. Ce main :

- génère, par appel à la fonction `generate_dot,` les fichiers `.dot` correspondants aux fichiers de données dont les noms ont été passés en paramètre.
- transforme ces fichiers `.dot` en `.pdf` par appel système à la commande `dot`.

La fonction `generate_dot` prépare le travail avec l'entête et la fin du fichier `.dot`. Elle traite également les cas limites où l'arbre est vide ou réduit à une feuille (donc sans arcs). Sinon elle appelle la fonction `recursive_dot` dans le cas général: génération récursive des arcs ``` ``valeur racine -> valeur fils;'' ``` (s'ils existent).

Travail à faire - 2 fonctions

1. Il reste à programmer la fonction `recursive_dot` qui traite le cas général.
2. Tester ensuite par:
   `bash> ./trees ../samples/*.txt`
   Vous devez retrouver les fichiers `.dot` et .`pdf` fournis.

[1] et plus généralement de graphes exploitables par des logiciels tels que http://www.graphviz.org

[2] `digraph` signifie "directed graph", les arbres en font partie du fait de leur orientation père → fils.