Structures de données et algorithmes

Contrôle TP du 20 décembre 2024 (CC3, durée : 1h30)

Arbres binaires de recherche

Notation. On rappelle que l'acronyme ABR abrège l'expression « arbre binaire de recherche ». Dans les fichiers fournis pour ce TP noté, les commentaires sont presque toujours rédigés en anglais, aussi y rencontre-t-on l'acronyme BST, qui abrège « *binary search tree* ».

1 Description du travail demandé

Le barème du contrôle de TP est provisoirement sur 25 points. Le détail des points accordés est donné dans la description du programme demandé (cf. les étapes 1 à 10 ci-dessous) et dans la description de la procédure de remise du travail (cf. section 2, page 2). **Aucun point ne sera accordé si le programme ne compile pas !**

Après avoir récupéré les quatorze (14) fichiers (dont celui que vous lisez) fournis dans le dossier « Fichiers pour le contrôle du 20 décembre 2024 », vous devez compléter le fichier `main_bst.c` (et accessoirement les fichiers `binary_tree.c` et `bst.c`) de sorte que le fichier `main_bst.c` contienne le code d'un programme qui exécute chacune des dix étapes suivantes :

1. lit en argument sur la ligne de commande deux entiers positifs n et m [1 pt] ;
2. initialise de façon pseudo-aléatoire deux chaînes de caractères constituées de lettres majuscules de l'alphabet latin, l'une de longueur n, l'autre de longueur m, et les affiche [2 pt] ;
3. construit un ABR de taille n en insérant successivement les valeurs de la première chaîne comme des feuilles et l'affiche [1] ; [1,5 pt si le programme fait appel à la fonction récursive `insert_BST` de la bibliothèque « `bst` », 3 pt s'il fait appel à la fonction `insert_BST_it`, version non récursive à compléter dans le fichier `bst.c`] ;
4. construit un ABR de taille m en insérant successivement les valeurs de la deuxième chaîne à la racine et l'affiche [1,5 pt si le programme fait appel à la fonction récursive `insert_BST_root` de la bibliothèque « `bst` », 3 pt s'il fait appel à la fonction `insert_BST_root_it`, version dérécursivée à compléter dans le fichier `bst.c`] ;
5. supprime de l'ABR de taille n la valeur de rang $\lfloor n/2 \rfloor$ en faisant appel aux fonctions `delete_node_BST` (à compléter dans le fichier `bst.c`) et `select_BST` et affiche l'ABR après suppression [3 pt] ;
6. fusionne les deux ABR et affiche l'ABR t résultant de la fusion en faisant appel à la fonction `join_BST` à compléter dans le fichier `bst.c` [3 pt] ;
7. affiche les étiquettes de l'ABR t dans l'ordre alphabétique [1,5 pt si le programme fait appel à la fonction récursive `traverse_inorder_binary_tree` de la bibliothèque « `binary_tree` », 3 pt s'il fait appel à la fonction `traverse_inorder_it_BT`, version dérécursivée à compléter dans le fichier `binary_tree.c`] ;
8. rééquilibre l'ABR t en faisant appel à la fonction `balance_BST` à compléter dans le fichier `bst.c` et affiche l'ABR rééquilibré [3 pt] ;

1. Pour l'affichage des arbres binaires demandé dans cette étape et dans les suivantes, vous utiliserez la fonction `show_binary_tree` de la bibliothèque « `binary_tree` ».

9. affiche les étiquettes de l'ABR t rééquilibré dans l'ordre résultant du parcours par niveau en faisant appel à la fonction `traverse_level_BT` à compléter dans le fichier `binary_tree.c` [2 pt] ;
10. libère tout l'espace mémoire alloué sur le tas [1 pt].

N.B.1. Pour compiler votre programme, vous pouvez utiliser le fichier `Makefile` fourni avec les autres fichiers.

N.B.2. Le fichier `prog_bst` est un exécutable que vous pouvez lancer (en lui passant deux arguments entiers positifs) pour avoir une idée de ce qui vous est demandé.

N.B.3. Les noms des fichiers ne doivent en aucun cas être modifiés. Vous ne pouvez modifier que les fichiers `binary_tree.c`, `bst.c` et `main_bst.c`, en substituant du code aux passages repérés par les commentaires `/** À COMPLÉTER **/` ou `/** TO BE COMPLETED **/`.

2 Procédure de remise du travail

Ajoutez en commentaire au début des fichiers `binary_tree.c`, `bst.c` et `main_bst.c` les nom, prénom et groupe de chacun(e) des étudiant(e)s ayant contribué au travail remis. Créez **dans votre répertoire personnel** un sous-répertoire appelé `CC3_SDA` et réunissez-y les onze (11) fichiers sources (`.h` et `.c`), le fichier Makefile et seulement ces fichiers. **Placez-vous dans votre répertoire personnel** et créez une archive compressée à partir du sous-répertoire `CC3_SDA` en lançant dans un terminal l'une des commandes suivantes

```
tar -a -cf cc3_SDA.zip ./CC3_SDA
tar -a -cf cc3_SDA.gz ./CC3_SDA
tar -a -cf cc3_SDA.gzip ./CC3_SDA
tar -a -cf cc3_SDA.tgz ./CC3_SDA
```

Déposez l'archive compressée `cc3_SDA.zip` (ou `cc3_SDA.gz`, `cc3_SDA.gzip`, `cc3_SDA.tgz`) de votre répertoire personnel dans la zone de dépôt de votre groupe de TP [1 pt si tous les fichiers sont présentés dans une archive compressée].

N.B. La zone de dépôt n'accepte qu'un fichier .zip, .gz, .gzip ou .tgz.