Feuille de Travaux Pratiques
Tableau d'ABR

Novembre-Décembre 2016

Ce projet est à réaliser en **binômes** (un seul monôme autorisé si le groupe possède un nombre impair d'étudiants). Six séances de TP d'1h20 y seront consacrées. La programmation se fera dans le langage de votre choix. *Assurez-vous que vos programmes compilent et fonctionnent sous Linux sur les machines du CIE.*
Vous rendrez l'ensemble des sources de votre projet, sous la forme d'une archive NOM1-NOM2.zip. La décompression de l'archive doit produire un répertoire NOM1-NOM2 contenant tous les éléments de votre travail (compte-rendu, sources, exécutable, jeux d'essai) et un fichier texte contenant les instructions détaillées de compilation et d'exécution.

Cette archive est à **déposer sous Madoc**. La date limite de restitution est fixée au **Vendredi 13 Janvier 2017 à 20h17**. Un malus sera appliqué lorsqu'une ou plusieurs des consignes données ci-dessus n'auront pas été respectées.

1 Tableau d'ABR (TABR)

Nous considérons dans ce projet une structure de données appelée *tableau d'ABR* (ou TABR). Il s'agit d'un tableau T dont chaque case i correspond à un intervalle $[T[i].debut; T[i].fin]$. Les informations stockées dans la case i d'un TABR sont $T[i].debut$, $T[i].fin$ et un pointeur $T[i].arbre$ vers un ABR **sans doublons**, dont tous les éléments se situent dans l'intervalle $[T[i].debut; T[i].fin]$. Les intervalles ont les propriétés suivantes :

(a) Ils sont bien définis : pour tout i, $T[i].debut \leq T[i].fin$;
(b) Ils sont tous disjoints : deux intervalles différents n'ont aucun élément en commun ;
(c) Ils sont ordonnés par ordre croissant : pour tout i (sauf le dernier), $T[i].fin < T[i+1].debut$.

Les extrémités $T[i].debut$ et $T[i].fin$ de chaque intervalle sont des entiers, et les éléments dans les ABR indiqués par les pointeurs $T[i].arbre$ sont également des entiers. La Figure 1 ci-dessous vous montre un exemple de TABR.

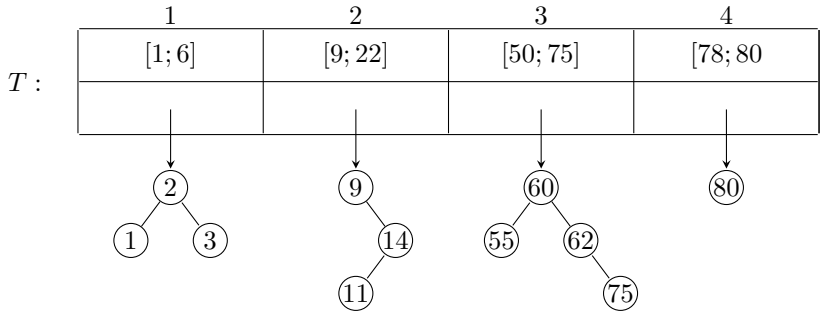

FIGURE 1 – Exemple de TABR

2 Travail demandé

1. Proposer un programme convivial qui, par l'intermédiaire d'un menu, permet à l'utilisateur de choisir de réaliser une ou plusieurs des fonctions détaillées ci-dessous (Sections 2.1 et 2.2).
   Remarque : pour toute création (décidée par vous) d'un ABR possédant au moins 3 éléments, l'arbre filiforme est interdit.
2. Pour chacune de ces fonctions, indiquer sa complexité au pire, en détaillant vos arguments dans votre compte-rendu. On attend ici une complexité sous la forme d'un $O()$ ou d'un $\Theta()$, en étant bien entendu le plus précis possible.

2.1 Génération, sauvegarde et affichage des TABRs

1. (**Fichier vers TABR**) Écrire une fonction permettant de créer un TABR T à partir d'un fichier f donné. Chaque ligne i du fichier f correspond à la case i de T, et le contenu d'une ligne i est une chaîne

   $$a : b\,;\, x_1 : x_2 : \ldots : x_k$$

   telle que :
   — $a = T[i].debut$ et $b = T[i].fin$
   — $x_1 x_2 \ldots x_k$ est le résultat du parcours *suffixe* de l'ABR $T[i].arbre$
   Par exemple, le contenu du fichier correspondant au TABR de la Figure 1 est :

   ```
   1:6;1:3:2
   9:22;11:14:9
   50:75;55:75:62:60
   78:80;80
   ```

2. (**TABR vers fichier**) Écrire une fonction qui effectue la sauvegarde d'un TABR dans un fichier, dont le nom et l'emplacement sont fournis par l'utilisateur. Le fichier doit bien entendu avoir le même format que ci-dessus.
3. (**Affichage sur l'écran**) Écrire une fonction permettant d'afficher un TABR à l'écran. Par "afficher un TABR" T, on entend reproduire à l'écran ce que contiendrait le *fichier* correspondant à T.
4. (**TABR aléatoire**) Écrire une fonction qui prend en argument deux entiers n et m et qui crée un TABR T de n cases remplies avec des données générées aléatoirement telles que, pour tout $1 \leq i \leq n$, on ait $T[1].debut = 1$, $T[n].fin = m$ et $T[i].arbre$ contient au moins 3 éléments.
5. (**Vérification**) Écrire une fonction qui vérifie si un TABR T construit à partir d'un fichier donné en argument est bien rempli, c'est-à-dire que :
   — les intervalles $[T[i].debut; T[i].fin]$ vérifient les propriétés (a), (b) et (c) décrites en Section 1, et
   — $T[i].arbre$ est un ABR dont les éléments sont des entiers contenus dans l'intervalle $[T[i].debut; T[i].fin]$.

2.2 Manipulation des TABRs

1. (**Insertion d'un entier**) Écrire une fonction qui réalise l'insertion d'un entier x dans un TABR T. Cela suppose de trouver l'indice i tel que $[T[i].debut; T[i].fin]$ contient x, et d'insérer x dans $T[i].arbre$. Si (1) aucun des intervalles $[T[i].debut; T[i].fin]$ ne contient la valeur x ou si (2) x existe déjà dans le TABR, l'insertion ne sera pas faite. On indiquera alors à l'écran dans quel cas ((1) ou (2)) on se trouve.
2. (**Suppression d'un entier**) Écrire une fonction de suppression d'un entier x d'un TABR. Cela suppose de trouver l'arbre $T[i].arbre$ dans lequel x se trouve et de supprimer x de ce $T[i].arbre$. Si x n'existe pas dans le TABR, la suppresion ne sera pas faite. On indiquera alors à l'écran dans quel cas on se trouve : (1) aucun intervalle ne contient x, ou (2) un des intervalles contient x, mais l'ABR correspondant à cet intervalle ne contient pas x. Dans le cas (2), on affichera l'indice i de l'intervalle qui contient x.
3. (**Fusion de deux cases du TABR**) Écrire une fonction qui, étant donné un indice $i < n$, transforme un TABR donné T en un TABR T' fusionné comme suit :
   — on fusionne les intervalles $[T[i].debut; T[i].fin]$ et $[T[i+1].debut; T[i+1].fin]$ en un seul intervalle $[T[i].debut; T[i+1].fin]$,
   — on fusionne les ABR $T[i].arbre$ et $T[i+1].arbre$ en un seul ABR (selon la méthode de votre choix, à décrire dans le compte-rendu),

— on supprime la case $i+1$ de T (ceci se fait en décalant toutes les cases de T qui se trouvent à droite de la case éliminée),
— on met à jour les informations dans la case i ($T[i].debut$, $T[i].fin$ et $T[i].arbre$ doivent correspondre au nouvel intervalle et au nouvel ABR).

4. (**Équilibre en Profondeur d'un ABR**) Écrire une fonction qui, étant donné un indice $1 \leq i \leq n$, indique si l'ABR stocké en $T[i].arbre$ est équilibré en profondeur. Si la réponse est NON, la fonction affichera à l'écran trois informations : (1) la valeur d'un nœud v pour lequel l'équilibre en profondeur n'est pas respecté, (2) la profondeur du sag de v et (3) la profondeur du sad de v. Note : par convention, la profondeur d'un arbre vide est égale à -1.

5. (**ABR vers TABR**) Soit un ABR A dont les éléments sont compris entre deux valeurs MIN et MAX. On veut construire un TABR contenant les éléments de A. Pour cela, on découpe l'intervalle [MIN ;MAX] en k intervalles notés $[a_1; b_1]$, $[a_2; b_2]$ ... $[a_k; b_k]$ tels que $a_1 =$MIN , $b_k =$MAX et l'union de $[a_1; b_1]$, $[a_2; b_2]$, ..., $[a_k; b_k]$ est exactement [MIN ;MAX].
   Écrire une fonction qui, étant donné un ABR A et les valeurs $b_1, b_2, \ldots b_{k-1}$ (A et les b_is étant fournis par l'utilisateur), construit le TABR qui correspond à la description ci-dessus. On sera très précis (à la fois dans le compte-rendu et dans l'interface) sur la façon dont l'utilisateur doit renseigner A.

6. (**TABR vers ABR**) Écrire une fonction qui, étant donné un TABR T, construit un ABR contenant l'ensemble des éléments des ABR contenus dans T (et aucun élément supplémentaire).