NF16-TP4 : Arbres binaires de recherche (ABR)

Ce TP est basé sur l'examen final de NF16 de l'Automne 2016

Objectif:
L'objectif de ce TP est de se familiariser avec les arbres binaires de recherche (ABR) et les différentes opérations nécessaires pour les manipuler.

Partie A : une représentation classique.

Soit E= { e_1, e_2, …, e_n } un ensemble de n entiers distincts. On souhaite représenter un tel ensemble en utilisant un arbre binaire de recherche dans lequel la clé de chaque sommet correspond à un élément de l'ensemble E.

1. Définissez la structure **`Sommet`** qui représente un sommet de l'arbre.
2. Définissez la structure **`Arbre`** qui représente un ABR.
3. Implémentez la fonction **`initABR`** qui renvoie un pointeur vers un ABR vide.
4. Implémentez la fonction **`creerSommet`** qui renvoie un pointeur vers un nouveau sommet à partir d'une clé entrée en paramètre.
5. Implémentez la fonction **`insererSommet`** qui insère un sommet dans un ABR. Rappel : on ne doit pas avoir deux fois la même clé dans notre ABR.
6. Implémentez la fonction **`afficherArbre`** qui affiche tous les élements de l'arbre triés par ordre croissant.
7. Implémentez la fonction **`rechercheSommet`** qui vérifie si une clé existe dans l'ABR.
8. Implémentez la fonction **`tailleArbre`** qui renvoie le nombre d'octets utilisés pour représenter l'ABR.

Partie B : une représentation plus compacte.

On souhaite maintenant représenter les ensembles d'entiers de manière plus compacte (en utilisant moins de mémoire) en profitant du fait que plusieurs éléments de l'ensemble sont consécutifs formant alors un intervalle de valeurs entières qui peut être représenté par sa borne inférieure et sa borne supérieure. Chaque Sommet x de l'arbre binaire de recherche représente alors un intervalle de valeurs entières consécutives (toutes appartenant à l'ensemble). Un sommet x comporte les informations :

- gauche[x], la racine du sous-arbre gauche de x (ou NULL si un tel sous-arbre n'existe pas)
- droit[x], la racine du sous-arbre droit de x (ou NULL si un tel sous-arbre n'existe pas)
- inf[x], la valeur inférieure de l'intervalle
- sup[x], la valeur supérieure de l'intervalle

Notons qu'une valeur e_i de l'ensemble qui n'est consécutive à aucun autre élément de l'ensemble E sera alors représentée par un sommet x où inf[x]=sup[x]=e_i.

Dans la suite, on ne s'intéressera qu'aux arbres où la propriété suivante est vérifiée :

Propriété : si x et y sont deux sommets d'un arbre A, alors soit sup[x]+1 < inf[y], soit sup[y]+1 < inf[x].

En effet, si la propriété n'est pas vérifiée pour deux sommets x et y, alors x et y peuvent être fusionnés en un seul sommet z avec inf[z]=min(inf[x], inf[y]) et sup[z]=max(sup[x], sup[y]).

1. Définissez la structure **`SommetCompact`** qui représente un sommet de l'arbre suivant cette nouvelle représentation.
2. Définissez la structure **`ArbreCompact`** qui représente un ABR suivant cette nouvelle représentation.
3. Implémentez la fonction **`initABRCompact`** qui renvoie un pointeur vers un ABR vide.
4. Implémentez la fonction **`creerSommetCompact`** qui renvoie un pointeur vers un nouveau sommet à partir d'un nombre entier entré en paramètre.
5. Implémentez la fonction **`insererElement`** qui insère un élément (un nombre entier) dans un ABR. Attention : lors de l'insertion d'un nouvel élément dans l'ABR, certains sommets doivent parfois être fusionnés pour respecter la propriété définie ci-dessus, pensez-donc à gérer toutes les situations possibles où cela doit être fait.
6. Implémentez la fonction **`afficherArbreCompact`** qui affiche tous les élements de l'arbre triés par ordre croissant.
7. Implémentez la fonction **`rechercheElement`** qui vérifie si un élément existe dans l'ABR.
8. Implémentez la fonction **`tailleArbreCompact`** qui renvoie le nombre d'octets utilisés pour représenter l'ABR.

Programme principal:

Vous proposerez un menu qui permettra :

- d'initialiser un ABR suivant l'une ou l'autre des deux représentations
- de pré-charger un ABR de manière aléatoire si l'utilisateur le souhaite
- de tester l'insertion dans l'ABR, l'affichage de l'ABR, la recherche dans l'ABR
- de comparer la taille occupée par l'ABR avec chacune des représentations

Consignes générales:

- L'organisation MINIMALE du projet sera la suivante:
  - Fichier d'en-tête tp4.h, contenant la déclaration des structures/fonctions
  - Fichier source tp4.c, contenant la définition de chaque fonction
  - Fichier source main.c, contenant le programme principal

- Votre rapport de quatre pages maximum contiendra :
  - La liste des structures et des fonctions supplémentaires que vous avez choisi d'implémenter et les raisons de ces choix.
  - Un exposé succinct de la complexité de chacune des fonctions implémentées.

Votre rapport et vos trois fichiers feront l'objet d'une remise de devoir sur Moodle dans l'espace qui sera ouvert à cet effet quelques jours suivant votre démonstration au chargé de TP (un seul rendu de devoir par binôme).