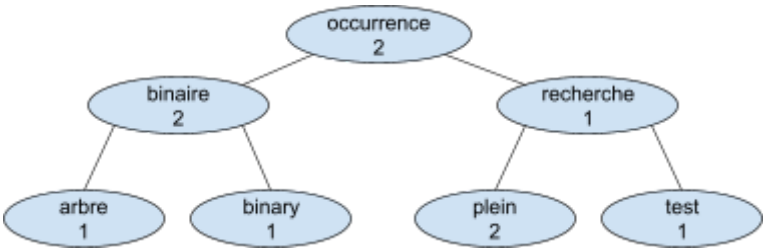
NF16 - TP4 : Arbre Binaire de Recherche

Le but de ce TP est de se familiariser avec les arbres binaires de recherche (ABR) et les différentes opérations nécessaires pour les manipuler. Nous souhaitons utiliser un arbre binaire de recherche pour indexer un texte, c'est-à-dire un arbre ordonné des mots composant ce texte comme clé avec le nombre d'occurrences de chaque mot dans le texte comme valeur.

Un nœud de l'arbre contient les informations suivantes **:**

- **`mot :`** un mot de type chaîne de caractères, c'est la clé du nœud
- **`nombre_occurences :`** le nombre d'occurrences dans le texte
- **`fils_droit :`** un pointeur vers le fils droit du nœud
- **`fils_gauche :`** un pointeur vers le fils gauche du nœud

La séquence de mots suivante "***occurrence binaire binary recherche test plein arbre binaire plein occurrence***" sera représentée sous la forme suivante :



B. Structures de données et types :

1. Définir la structure Nœud qui représente un nœud de l'ABR et le type correspondant T_Noeud.

2. Définir le type T_Arbre, un pointeur vers un nœud.

C. Fonctions à implémenter :

1. Écrire une fonction qui permet de créer un nœud et renvoie un pointeur vers ce nœud.

2. Écrire une fonction qui permet d'ajouter un nœud dans l'arbre et renvoie un pointeur vers la racine de l'arbre modifié.

NB :

- Avant d'ajouter un mot, on doit le convertir en minuscules. On supposera que l'utilisateur n'entre que des chaînes de caractères constituées de lettres (pas de chiffres ni de ponctuation).
- Si le mot est déjà présent dans l'arbre, on n'ajoute pas un nouveau nœud, mais on incrémente le nombre d'occurrences de ce mot.

3. Écrire une fonction qui permet de retirer une occurrence d'un mot d'un arbre et renvoie un pointeur vers la racine de l'arbre modifié.

Attention : si le nombre d'occurrences est à zéro, le nœud devra être supprimé de l'arbre. On veillera à libérer l'espace mémoire qui n'est plus utilisé.

4. Écrire une fonction qui permet d'afficher l'arbre.

   L'affichage se fera sous la forme suivante :

   A-- arbre [1]

   B-- binaire [2]

   binary [1]

   0-- occurrence [2]

5. Écrire une fonction qui renvoie si un ABR est est parfait.

NB :

- Un arbre binaire parfait est un arbre binaire dans lequel toutes les feuilles sont à la même hauteur dans l'arbre et chaque sommet interne a exactement deux fils.

6. Écrire une fonction qui renvoie si un ABR est équilibré.

NB :

- Un ABR est dit équilibré si pour chaque nœud de l'arbre, la différence de hauteur de ses sous-arbres en valeur absolue est au maximum un.

7. Écrire une fonction qui permet de transformer un arbre en un lexique en liste linéairement chaînée (vu en TP 3) et renvoie le pointeur vers cette liste.

   Un lexique, c'est une liste ordonnée des mots composant un texte.

8. Écrire une fonction qui permet de déterminer la similarité entre deux textes selon l'indice de Jaccard.

NB :

- L'indice de Jaccard est utilisé pour déterminer la similarité entre deux textes, c'est-à-dire combien de mots communs existent par rapport au nombre total de mots.

Exemple : text1 = {"arbre", "binaire", " de", "recherche"}, text2={"arbre", " de", "recherche"}

$$J = \frac{|text1 \cap text2|}{|text1 \cup text2|} = \frac{3}{4} = 0.75$$

D. Interface :

Mettre en place une interface permettant de tester vos fonctions en proposant le menu suivant :

1. Créer un nouvel arbre
2. Afficher un arbre
3. Ajouter un mot dans un arbre
4. Retirer un mot d'un arbre
5. Vérifier si un arbre est parfait
6. Vérifier si un arbre est équilibré
7. Transformer un arbre en un lexique en liste linéairement chaînée et afficher le lexique
8. Tester la similarité entre deux textes
9. Quitter

Important : Votre programme doit permettre de gérer plusieurs arbres.

Une attention particulière devra être accordée à la qualité de votre interface afin de faciliter les saisies utilisateur et l'utilisation des différentes fonctionnalités.

Consignes générales :

Sources

À la fin du programme, les blocs de mémoire dynamiquement alloués doivent être proprement libérés.

Rapport

Votre rapport de quatre pages maximums 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 fichiers feront l'objet d'une remise de devoir sur **Moodle** dans l'espace qui sera ouvert à cet effet (un seul rendu de devoir par binôme) après la démonstration au chargé de TP.