*Feuille d'exercices no. 4*

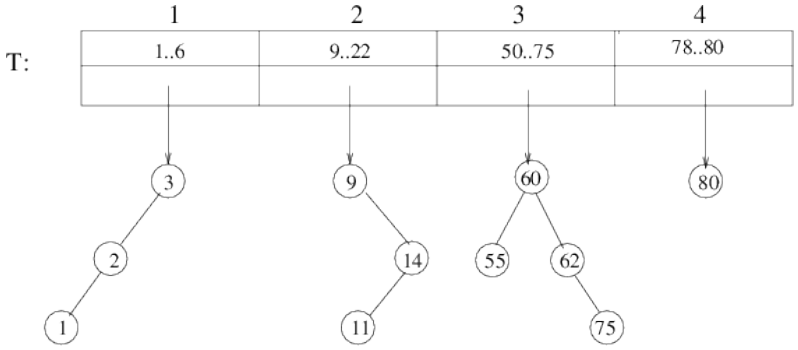
# Arbres binaires de recherche

**Exercice 1**. Soit un arbre initialement vide.

1. Construire un ABR par insertions successives des entiers suivants : 12, 8, 9, 14, 16, 13, 11, 10, 7.
2. Ensuite, trouver l'ABR obtenu après suppression successive de 11, 14 et 12.

**Exercice 2**. Nous considérons dans cet exercice une structure de données appelée *table d'arbres binaires de recherche* (abrégée TABR). Il s'agit d'un tableau $T$ dont chaque case $i$ correspond à un intervalle fermé $T[i].debut..T[i].fin$. Les informations stockées dans la case $i$ sont $T[i].debut$, $T[i].fin$ et un pointeur $T[i].arbre$ vers un arbre binaire de recherche dont tous les éléments se situent dans l'intervalle fermé $T[i].debut..T[i].fin$. Les intervalles ont les propriétés suivantes :
- ils sont bien définis, c.à.d. que pour tout $i$, $T[i].debut \leq T[i].fin$ ;
- ils sont tous disjoints, c.à.d. que deux intervalles différents n'ont aucun élément en commun ;
- ils sont ordonnés par ordre croissant, c.à.d. que pour tout $i$ (sauf le dernier), $T[i].fin < T[i+1].debut$.

Les extrémités $T[i].debut, T[i].fin$ de chaque intervalle sont des entiers, et les éléments dans les arbres binaires de recherche indiqués par les pointeurs $T[i].arbre$ sont également des entiers.




**On demande :**

1. Ecrire une fonction qui réalise l'insertion d'un entier $x$ dans une TABR appelée $T$. Cela suppose de trouver l'indice $i$ tel que $T[i].debut..T[i].fin$ contienne $x$, et d'insérer $x$ dans $T[i].arbre$. On supposera qu'au moins l'un des intervalles $T[i].debut..T[i].fin$ contient la valeur $x$ et que l'insertion est faite seulement si $x$ n'existe pas déjà dans la TABR.
2. Ecrire une procédure ou fonction de suppression d'un entier $x$ d'une TABR. Cela suppose de trouver l'arbre $T[i].arbre$ dans lequel $x$ se trouve (si $x$ est effectivement dans la TABR) et de supprimer $x$ de ce $T[i].arbre$. Si jamais $x$ n'existe pas dans la TABR, la procédure indiquera lequel des deux cas suivants est vérifié : soit aucun intervalle ne contient $x$ ; soit au moins l'un des intervalles contient $x$, mais l'arbre binaire de recherche ne contient pas $x$. Dans ce dernier cas, on retournera l'indice $i$ de l'intervalle qui contient $x$.
3. Ecrire une procédure ou fonction qui transforme une TABR donnée $T$ en une TABR $T'$ comme suit. On suppose que, en plus de $T$, nous disposons de l'indice $i$ d'un intervalle de $T$ (on supposera que $i$ est un indice arbitraire différent du dernier indice de $T$). Alors $T'$ est obtenue à partir de $T$ :
   - en fusionnant les intervalles $T[i].debut..T[i].fin$ et $T[i+1].debut..T[i+1].fin$ en un seul intervalle $T[i].debut..T[i+1].fin$,
   - en fusionnant les arbres $T[i].arbre$ et $T[i+1].arbre$ en un seul arbre binaire de recherche noté $P$,
   - en supprimant 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),
   - en mettant 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 arbre).
4. Soit un arbre binaire de recherche $A$ dont les éléments sont compris entre deux valeurs $Min$ et $Max$. On veut construire une TABR contenant les éléments de A. Pour cela, nous découpons l'intervalle $Min..Max$ en $k$ intervalles disjoints notés $a_1..b_1, a_2..b_2, \ldots, a_k..b_k$ tels que $a_1 = Min$ , $b_k = Max$ et la réunion de $a_1..b_1, a_2..b_2, \ldots, a_k..b_k$ est exactement $Min..Max$. On supposera que $a_1..b_1, a_2..b_2, \ldots, a_k..b_k$ sont déjà ordonnés par ordre croissant. Ecrire une fonction qui, étant donnés l'arbre binaire de recherche $A$ et les intervalles $a_1..b_1$, $a_2..b_2$, …, $a_k..b_k$, construit la TABR qui contient les intervalles donnés et les éléments de A.
5. Ecrire une procédure ou une fonction qui, étant donnée une TABR, construit un arbre binaire de recherche contenant tous les éléments de la TABR (et aucun élément supplémentaire).

**Exercice 3**. Dans un arbre binaire de recherche, on peut ajouter un élément non seulement aux feuilles mais aussi à la racine. Pour ajouter un élément $X$ à la racine d'un arbre binaire de recherche, il faut d'abord couper l'arbre de recherche initial en deux arbres binaires de recherche notés $G$ (contenant tous les éléments inférieurs ou égaux à $X$) et $D$ (contenant tous les éléments supérieurs à $X$), puis former le nouvel arbre binaire de recherche en utilisant $X$, $G$ et $D$.

1. Décrire, en français, un algorithme réalisant la coupure d'un arbre binaire de recherche $A$ en deux sous-arbres $G$ et $D$ selon un élément donné $X$. Bien indiquer les étapes à suivre.
2. Appliquer l'algorithme indiqué sur l'arbre $A$ ci-dessous, utilisant l'élément $X = 66$.
3. Ecrire une procédure $coupure(A, X, G, D)$ qui réalise la coupure d'un arbre binaire de recherche $A$ en deux arbres binaires de recherche $G$ et $D$ selon un élément donné $X$ (l'algorithme doit suivre les étapes présentées en 1)).
4. En utilisant $coupure$, écrire l'algorithme $ajoutracine$ qui ajoute un élément $X$ à la racine d'un arbre binaire de recherche (l'arbre obtenu doit être un arbre binaire de recherche).

**Pour aller plus loin ...**

**Exercice 4**. Ecrire une fonction qui prend en entrée un arbre dont chaque noeud contient un entier, et qui teste si c'est un arbre binaire de recherche ou non.