Travaux Dirigés
séance nr. 7

Arbres binaires de recherche équilibrés de type AVL

Partie 1

On définit un type "abint" pour un arbre binaire trié d'entiers. On conserve en chaque nœud la hauteur de cet arbre:

```
typedef struct s_abint = {
       T                    etiq;
       unsigned int         h;
       struct s_abint       *ag, *ad;
} *abint ;
```

1. Implanter les opérations : nouveau, enracinement, racine, fils gauche, fils droit, et vide.

2. Écrire une fonction « recherche » qui détermine si un élément appartient à un arbre binaire trié. Cette fonction renvoie l'arbre dont la racine est l'élément recherché ou un arbre NUL si l'élément n'est pas trouvé.

3. Écrire une fonction qui détermine si un arbre binaire donné est bien un arbre binaire trié : *est_trie(abint)*. Cette fonction renvoie un booléen. Elle ne vérifie pas l'équilibrage.

4. Écrire une fonction qui affiche les éléments de l'arbre par ordre croissant : affiche(abint)

Partie 2

Définition: Un arbre binaire de recherche est dit équilibré (ou bien balancé, en anglais *balanced tree*, b-tree) si en tout nœud les hauteurs des sous-arbres gauche et droit diffèrent au plus de un. On peut définir un indicateur de déséquilibrage:

```
entier deseq(abin *a)
{
si a==NIL alors renvoyer 0
sinon renvoyer hauteur(a->ag)-hauteur(a->ad)
}
```

alors un **arbre équilibré** est tel qu'en chacun de ses sous-arbres *b* on a: deseq(b)={-1, 0, 1}

On peut définir une rotation simple gauche ou droite ainsi qu'une rotation double gauche-droite ou droite-gauche. Toutes ces rotations transforment un arbre binaire de recherche en un arbre binaire de recherche. Une rotation double consiste en fait en une rotation d'un sous-arbre suivi d'une rotation de l'arbre. Les figures 1 et 2 illustrent deux exemples de rotations.

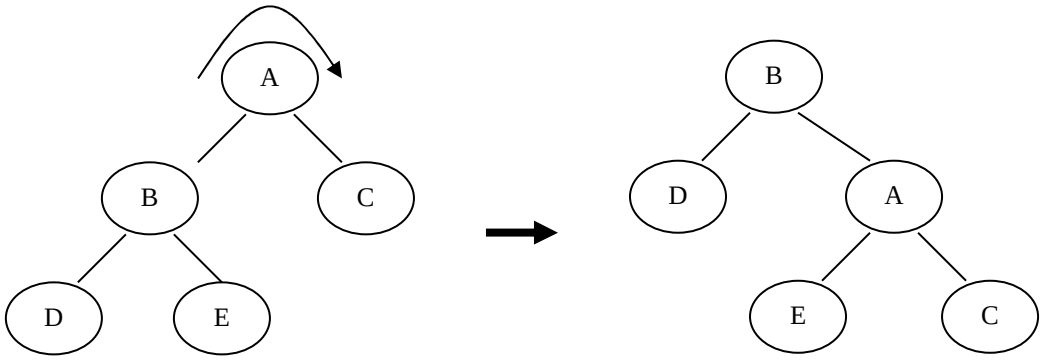
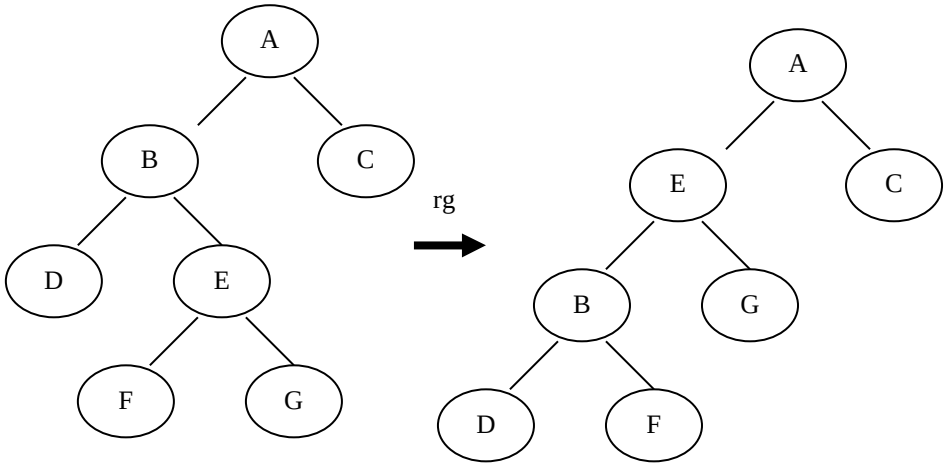


Figure 1: rotation droite (rd)



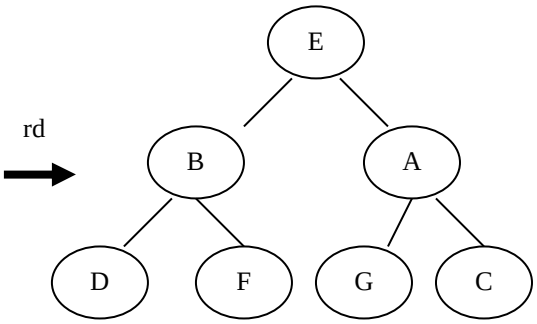


Figure 2: double rotation gauche-droite (rgd)

5. Ecrire les opérations de rotation et double rotation : *rg, rd, rgd, rdg*, sans oublier de mettre à jour les hauteurs stockées dans le champ *h*.

Un arbres de type Adelson – Velskii & Landis (AVL) est un arbre binaire de recherche toujours maintenu équilibré après chaque opération d'insertion ou de suppression. Un AVL est donc un arbre binaire de recherche équilibré.

6. Ecrire la fonction qui permet de faire l'insertion d'un élément en préservant l'équilibrage, écrire préalablement la fonction de mesure d'équilibrage *deseq* et une fonction *reeq* qui permet de faire un équilibrage. On utilisera une insertion en feuille.

Le rééquilibrage est nécessaire dans les cas de figure suivants (la figure montre le cas d'une insertion à gauche, le cas de l'insertion à droite est identique par symétrie):

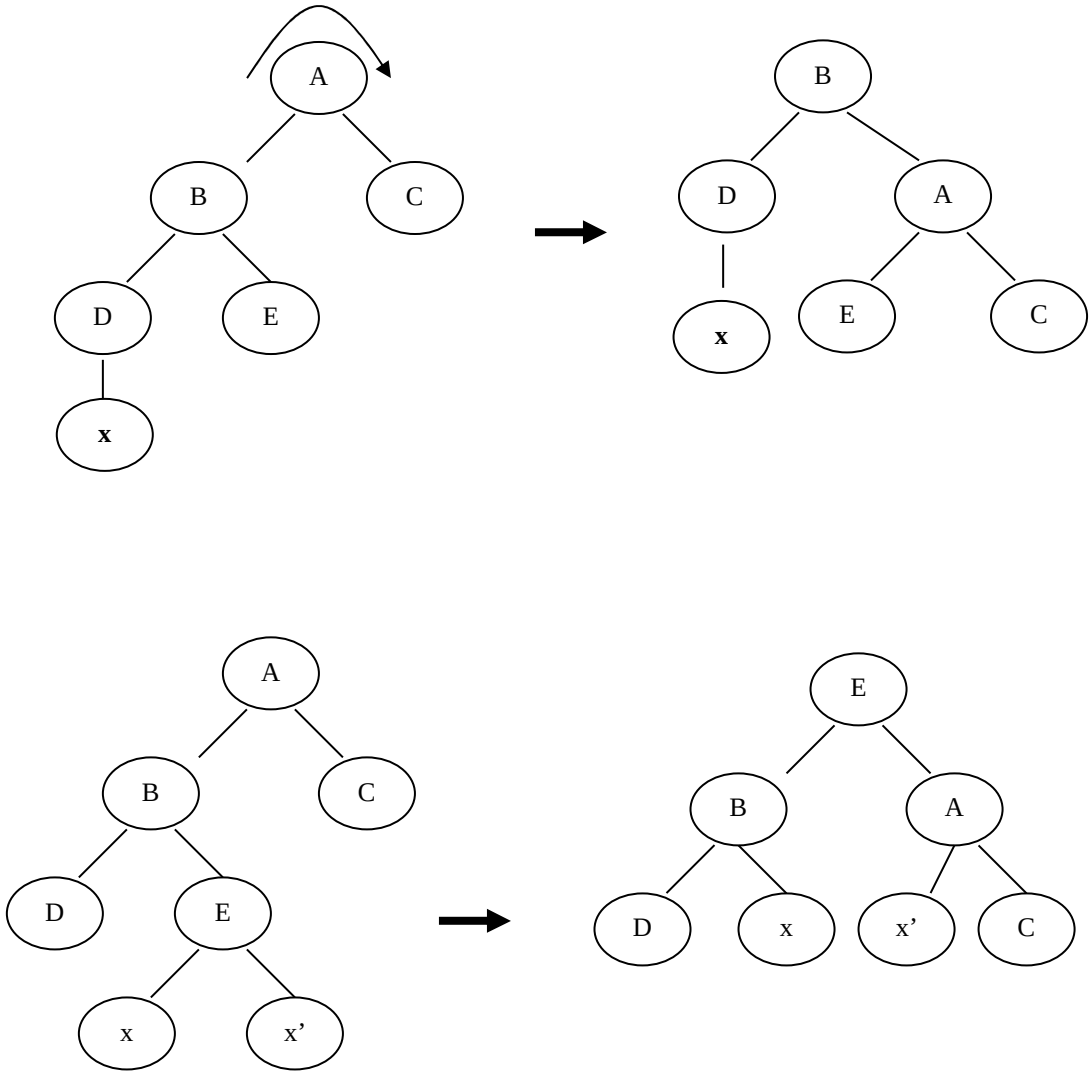


Figure 3: ici, x ou x' est ajouté en feuille dans a1 (sous arbre gauche), puis un rééquilibrage est effectué. Seuls ces deux cas nécessitent un rééquilibrage.