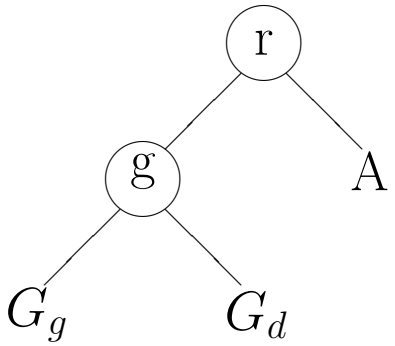
Travaux Dirigés d'algorithmique

Arbres binaires de recherche équilibrés AVL (Adelson-Velskii et Landis).

On notera $h(T)$ la hauteur de l'arbre T. Si G est le fils gauche de T et D le fils droit de T, on note $balance(T) = h(D) - h(G)$ la balance de l'arbre T.

1. **Manipulations.**

   (a) Effectuer respectivement une rotation droite sur la racine de l'arbre suivant, puis une rotation gauche sur la racine de l'arbre résultant:



   (a) Supposons que, dans l'arbre ci-dessus, les sous-arbres A, G_g, G_d soient équilibrés.

      i. Que dire du sous-arbre dont la racine est g si $|balance(g)| \leq 1$.

   Dans la suite, on supposera cette propriété vérifiée et on supposera que la balance de l'arbre, dont la racine est r, est -2;

      i. effectuer les rotations nécessaires afin d'équilibrer l'arbre ci-dessus dans le cas où la balance du sous-arbre dont le nœud g est la racine vaut 0 ou -1;
      ii. effectuer les rotations nécessaires afin d'équilibrer l'arbre ci-dessus dans le cas où la balance du sous-arbre dont le nœud g est la racine vaut 1;
      iii. calculer la balance de chacun des sous-arbres de l'arbre obtenu après la/les rotations(s) effectuée(s);

   (b) construire l'AVL par insertion successive des entiers suivants : $7, 12, 25, 16, 21, 13, 2, 5, 4, 9, 30, 27, 6, 11$. On dessinera les arbres avant et après les ajouts ayant entrainés une rotation, en indiquant les balances.

2. **Programmations.**
Dans tout cet exercice, on utilisera la structure suivante pour définir un AVL:

```
typedef struct noeud
{
    int valeur;
    int balance;
    struct noeud *fg, *fd;
} Noeud, *AVL;
```

- Ecrire une fonction `int estAVL(AVL a)` qui teste si l'arbre binaire de recherche `a` est un arbre binaire de recherche équilibré (*on suppose que a été construit indépendemment, le champs* `balance` *n'est pas rempli*);
- écrire une fonction `void RotationD(AVL *a)` qui effectue une rotation droite de l'arbre `a`. On n'oubliera pas de rectifier la balance des nœuds concernés;
- écrire une fonction `void RotationG(AVL *a)` qui effectue une rotation gauche de l'arbre `a`. On n'oubliera pas de rectifier la balance des nœuds concernés;
- écrire une fonction `void RotationGD(AVL *a)` qui effectue une rotation gauche/droite de l'arbre `a`. On n'oubliera pas de rectifier la balance des nœuds concernés. Une rotation gauche/droite est la double rotation effectuée à la question `2.c` de l'exercice 1;
- écrire une fonction `void Equilibre(AVL *a)` qui choisit et appelle les fonctions de rotation afin d'équilibrer l'arbre `a`. On supposera que les fils de `a` sont des AVL;
- écrire une fonction `void Insertion(AVL *a, int x)` qui effectue l'insertion de l'entier `x` dans l'AVL `a`. On effectuera les modifications de la balance des nœuds ainsi que les rotations, si nécessaire, afin de conserver la structure d'arbre AVL.