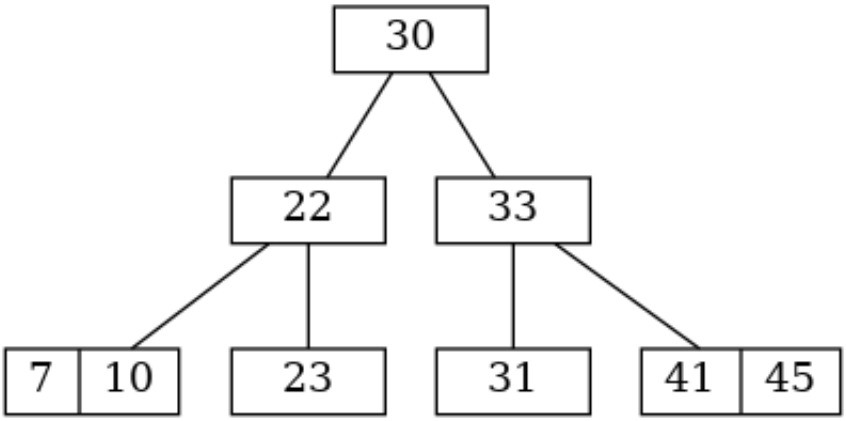
Algorithmique
Correction Partiel n° 2 (P2)

Info-sup S2 – Epita

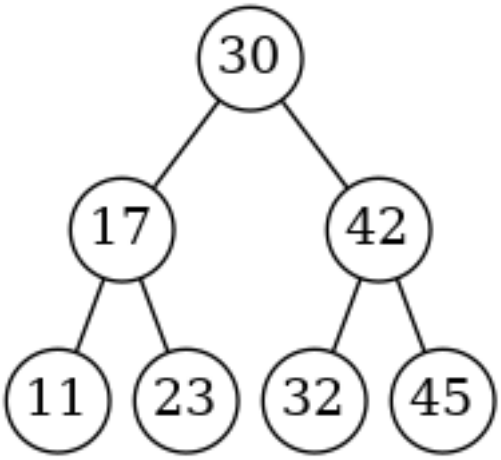
Solution 1 **(Arbre 2-3-4 : Ajout -** ***2 points*****)**



Solution **(Dessins – *4 points*)**

1. *Insertions*

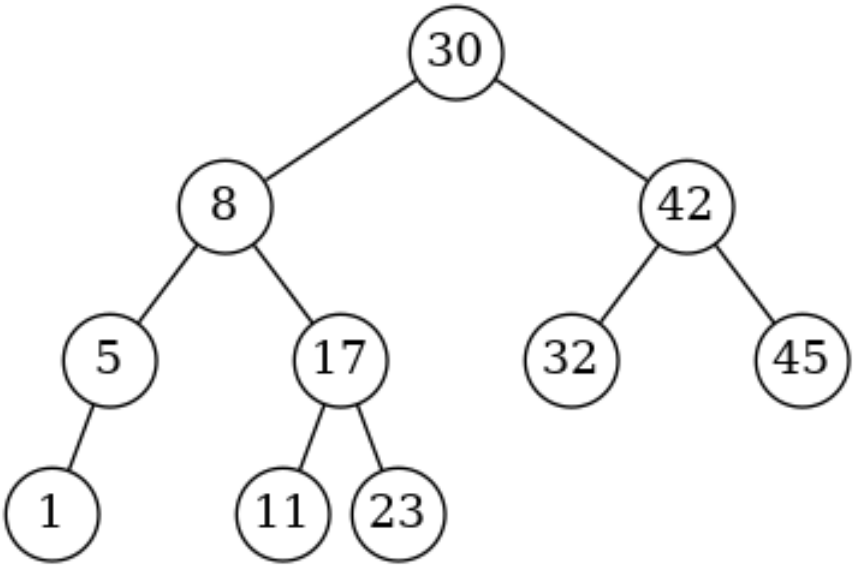
Arbre créé par insertions de 11, 30, 42, 32, 45, 23, 17 :



Rotations :

lr(11) / rg(11)
rlr(11) / rdg(11)

Arbre après ajouts de 5, 8, 1 :

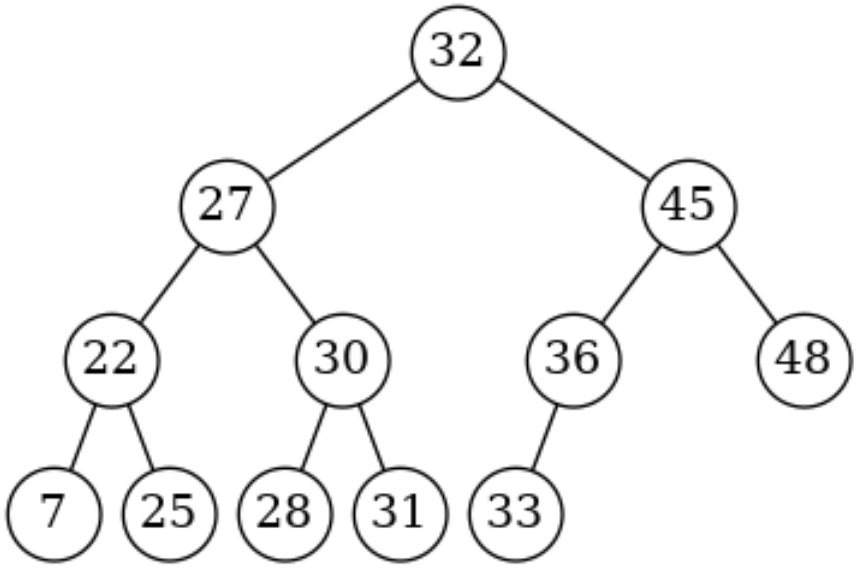


Rotations :

lrr(11) / rgd(11)
rr(17) / rd(17)

2. *Suppression*

Arbre après suppression de 21 :



Rotations :

rlr(7) / rdg(7)
lr(27) / rg(27)

***Solution 3* (Ajout avec mise à jour de la taille – *5 points*))**

Spécifications :

La fonction `addwithsize(`B`,` x`)` ajoute x en feuille dans l'arbre binaire de recherche B (`BinTreeSize`) sauf si celui-ci est déjà présent. Elle retourne un couple : (l'arbre résultat, un booléen indiquant si l'insertion a eu lieu).

```
    def addwithsize(B, x):
        if B == None:
            return BinTreeSize(x, None, None, 1), True
        else:
            if B.key == x:
                return B, False
            else:
                if x > B.key:
                    B.right, res = addwithsize(B.right, x)
                    if res:
                        B.size = B.size + 1
                else:
                    B.left, res = addwithsize(B.left, x)
                    if res:
                        B.size = B.size + 1
                return B, res
```

***Solution 4* (Plus proche ancêtre commun – *5 points*)**

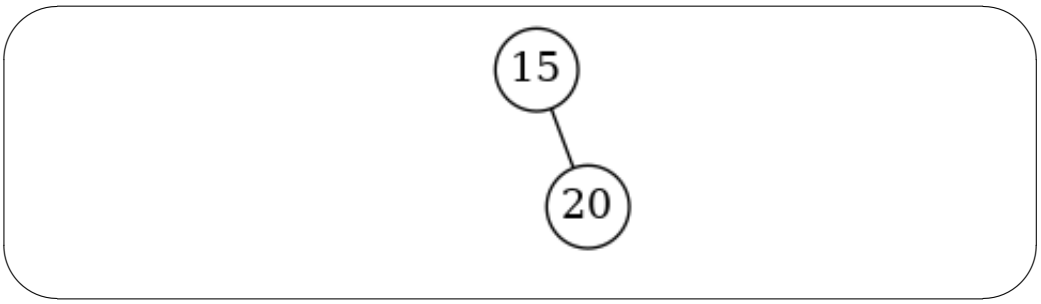
Spécifications :

La fonction `lca(`B`,` x`,` y`)` retourne la clé du plus proche ancêtre commun des noeuds contenant x et y dans l'arbre binaire de recherche B (avec $x \neq y$ tous les 2 présents dans B).

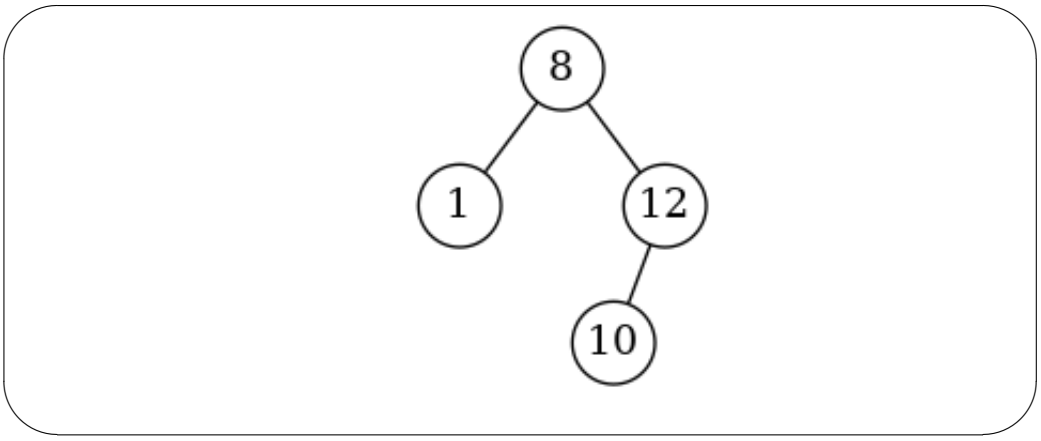
```
    def aux_lca(B, x, y):
        if B.key < x:
            return aux_lca(B.right, x, y)
        else:
            if B.key > y:
                return aux_lca(B.left, x, y)
            else:
                return B.key

    def lca(B, x, y):
        if x > y:
            x, y = y, x
        return aux_lca(B, x, y)
```

Solution 5 **(Mystery –** ***4 points*****)**

1. Arbre résultat de `mystery(`$B1$`, 13, 22)`



2. Arbre résultat de `mystery(`$B1$`, −2, 13)`



3. **Spécifications :**
   Implémenter l'opération `mystery(B, x, y)` en Python avec B arbre binaire de recherche et x et y deux entiers tels que $x < y$ qui retourne l'arbre modifié.

```
        def mystery(B, x, y):
            if B == None:
                return None
            else:
                if B.key < x:
                    return mystery(B.right, x, y)
                elif B.key > y:
                    return mystery(B.left, x, y)
                else:
                    B.left = mystery(B.left, x, y)
                    B.right = mystery(B.right, x, y)
                    return B

```