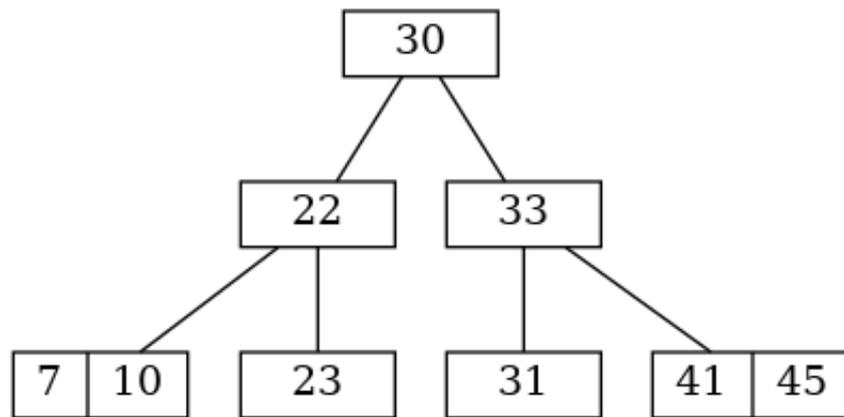


Algorithmique

Correction Partiel n° 2 (P2)

INFO-SUP S2 – EPITA

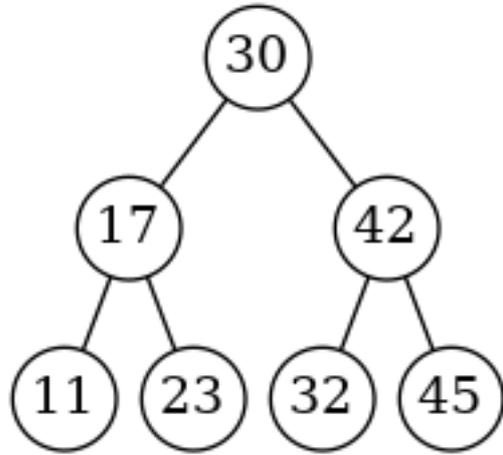
Solution 1 (Arbre 2-3-4 : Ajout - 2 points)



Solution (Dessins – 4 points)

1. Insertions

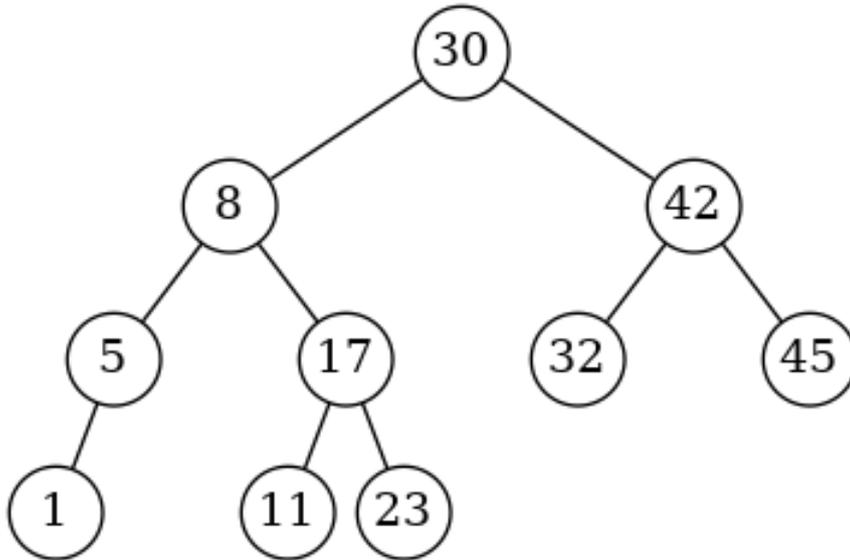
Arbre créé par insertions de 11, 30, 42, 32, 45, 23, 17 :



Rotations :

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

Arbre après ajouts de 5, 8, 1 :

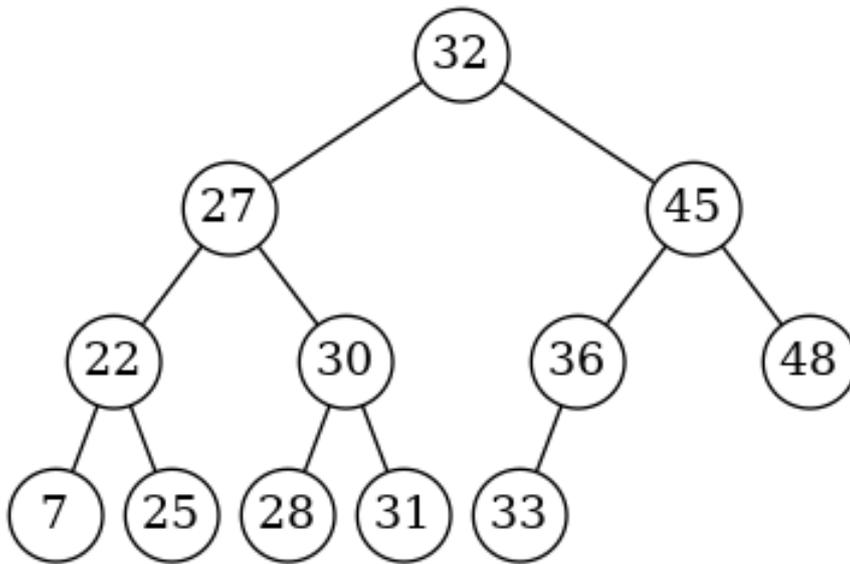


Rotations :

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).

```
1 def addwithsize(B, x):
2     if B == None:
3         return BinTreeSize(x, None, None, 1), True
4     else:
5         if B.key == x:
6             return B, False
7         else:
8             if x > B.key:
9                 B.right, res = addwithsize(B.right, x)
10                if res:
11                    B.size = B.size + 1
12            else:
13                B.left, res = addwithsize(B.left, x)
14                if res:
15                    B.size = B.size + 1
16            return B, res
```

Solution 4 (Plus proche ancêtre commun – 5 points)

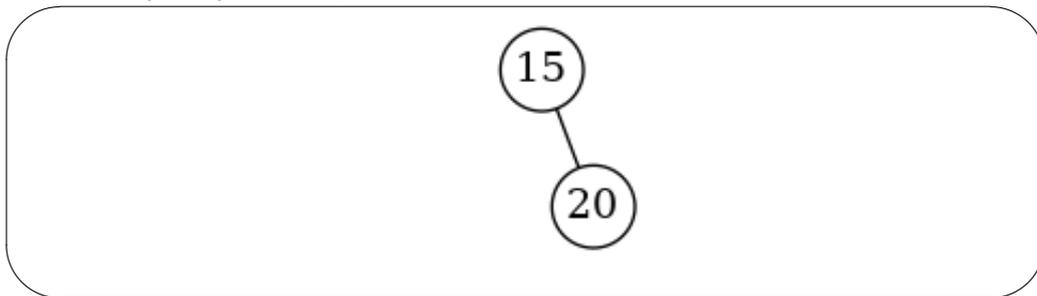
Spécifications :

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).

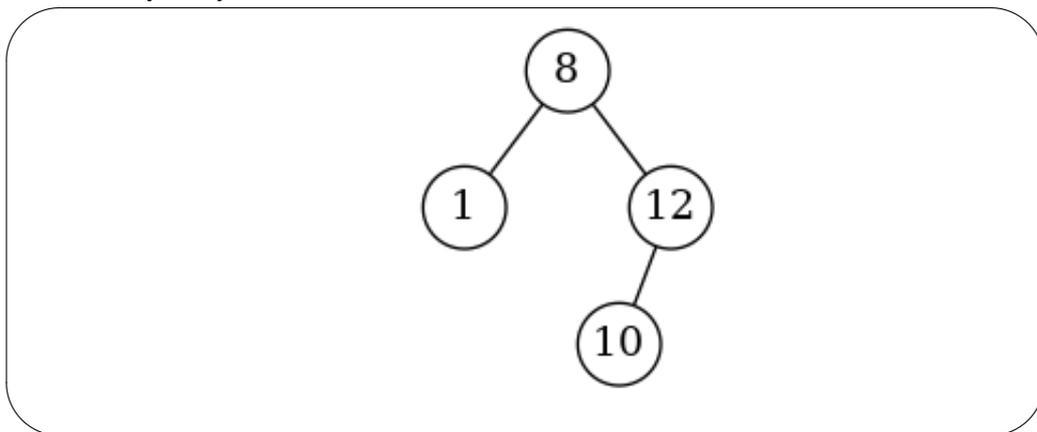
```
1 def aux_lca(B, x, y):
2     if B.key < x:
3         return aux_lca(B.right, x, y)
4     else:
5         if B.key > y:
6             return aux_lca(B.left, x, y)
7         else:
8             return B.key
9
10 def lca(B, x, y):
11     if x > y:
12         x, y = y, x
13     return aux_lca(B, x, 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.

```
1 def mystery(B, x, y):
2     if B == None:
3         return None
4     else:
5         if B.key < x:
6             return mystery(B.right, x, y)
7         elif B.key > y:
8             return mystery(B.left, x, y)
9         else:
10            B.left = mystery(B.left, x, y)
11            B.right = mystery(B.right, x, y)
12            return B
13
```