Exercices ABR - correction

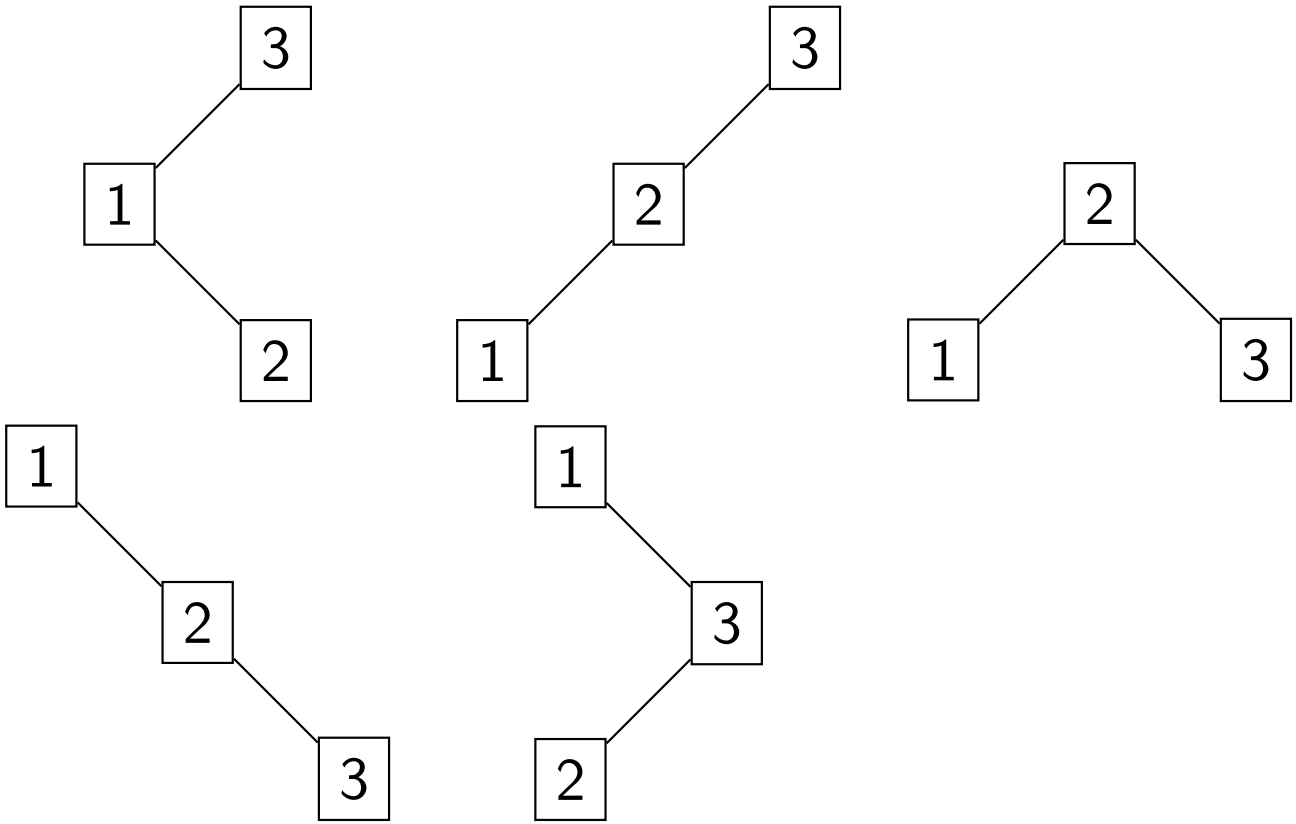
Christophe Viroulaud

Terminale - NSI

Algo 10

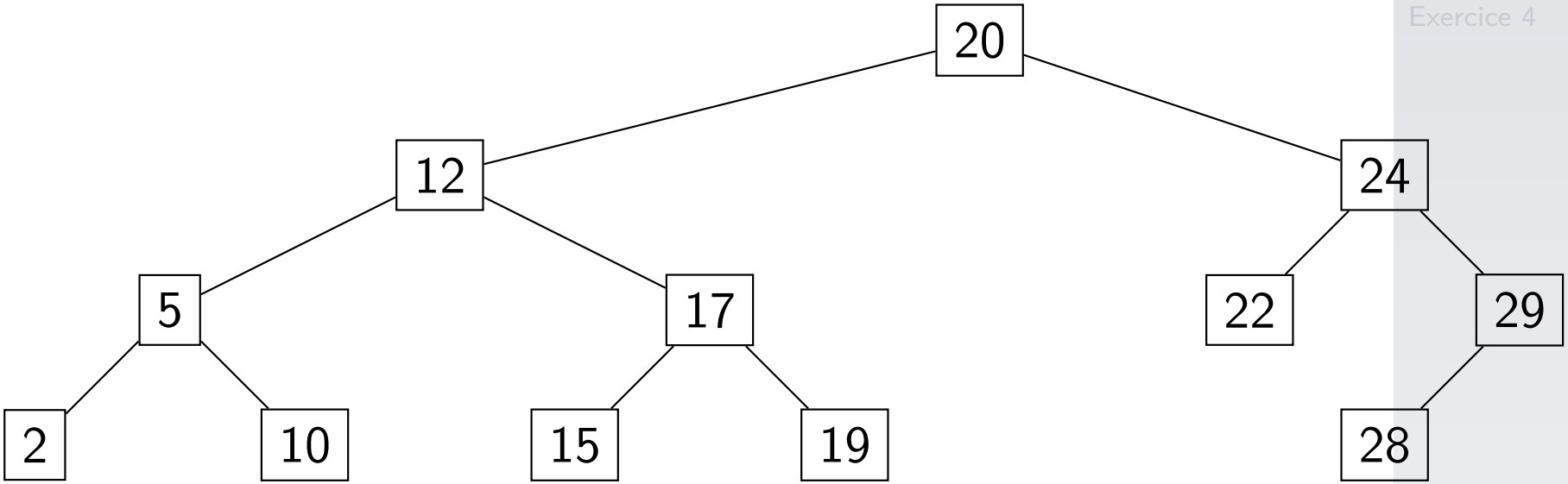
Sommaire

Exercice 1



Sommaire

Exercice 2



[2, 5, 10, 12, 15, 17, 19, 20, 22, 24, 28, 29]

Sommaire

Exercice 3


```
1  class Noeud:
2      def __init__(self, v: int):
3          self.valeur = v
4          self.gauche = None
5          self.droit = None
```

```python
def inserer(self, v: int) -> None:
    if v < self.valeur:  # gauche
        if self.gauche is None:  # feuille
            self.gauche = Noeud(v)
        else:  # descente récursive
            self.gauche.inserer(v)
    else:  # droite
        if self.droit is None:  # feuille
            self.droit = Noeud(v)
        else:  # descente récursive
            self.droit.inserer(v)
```

```
1  arbre = Noeud(13)
2  arbre.inserer(29)
3  arbre.inserer(2)
4  arbre.inserer(49)
5  arbre.inserer(8)
6  arbre.inserer(12)
7  arbre.inserer(16)
8  arbre.inserer(30)
9  arbre.inserer(27)
10 arbre.inserer(10)
11 arbre.inserer(9)
```

```python
def rechercher(self, v: int) -> None:
    if v == self.valeur: # trouvé
        return True
    elif v < self.valeur:  # gauche
        if self.gauche is None:  # feuille
            return False
        else:  # descente récursive
            return self.gauche.rechercher(v)
    else:  # droite
        if self.droit is None:  # feuille
            return False
        else:  # descente récursive
            return self.droit.rechercher(v)
```

```
1 print(arbre.rechercher(16))  # True
2 print(arbre.rechercher(17))  # False
```

```
1  def minimum(self) -> int:
2      n = self # instance en cours
3      while n.gauche is not None:
4          n = n.gauche
5      return n.valeur
6
7
8  def minimum_rec(self) -> int:
9      if self.gauche is None:
10         return self.valeur
11     else:
12         return self.gauche.minimum_rec()
```

```
1 def infixe(self, parcours: list) -> None:
2     if self.gauche is not None:
3         self.gauche.infixe(parcours)
4     parcours.append(self.valeur)
5     if self.droit is not None:
6         self.droit.infixe(parcours)
```

Sommaire

Exercice 4


```python
def taille(a: dict, s: str) -> int:
    if a[s][0] == '' and a[s][1] == '':  # pas de fils
        return 1
    elif a[s][0] == '':  # pas de fils gauche
        return 1 + taille(a, a[s][1])
    elif a[s][1] == '':  # pas de fils droit
        return 1 + taille(a, a[s][0])
    else:  # deux fils
        return 1 + taille(a, a[s][1]) + taille(a, a[s][0])
```