

**Exercice 1 : Dichotomie**

1. Rappeler le programme impératif de la recherche dichotomique.
2. Écrire une version récursive de cet algorithme. Il sera peut-être nécessaire d'ajouter des arguments.
3. Créer une liste  $l$  de cinquante éléments triés.
4. Tester les deux fonctions sur la liste  $l$ .
5. Rappeler la complexité de l'algorithme de recherche dichotomique.

**Exercice 2 :** Le tri rapide est un autre exemple d'algorithme utilisant la méthode *diviser pour régner*. C'est un algorithme naturellement récursif que nous pouvons décrire ainsi :

- Choisir un élément pivot.
- Sélectionner tous les éléments inférieurs au pivot.
- Sélectionner tous les éléments supérieurs ou égaux au pivot.
- Placer récursivement à gauche du pivot les éléments inférieurs à ce-dernier et à droite les éléments supérieurs.

- $\div$  : partitionner en 3 (gauche, pivot, droite)
- régner : les 2 sous-tableaux sont traités récursivement
- combiner : pas de recombinaison : les tableaux ont été triés en place

1. Écrire une fonction **tri\_rapide(tab : list) list** qui implémente l'algorithme du tri rapide. Nous choisirons le premier élément du tableau comme pivot.
2. Construire en compréhension une liste  $l$  de vingt éléments compris entre 0 et 100.
3. Tester la fonction de tri sur la liste  $l$ .

**Remarque :** Le tri rapide a une complexité en  $O(n \times \log_2(n))$ .

**Exercice 3 :** La fonction *mystere* implémente un algorithme du type *diviser pour régner*.

```
1 def mystere(tab: list, debut: int = 0, fin: int = -1) -> int:
2     #initialise 'fin'
3     if fin == -1:
4         fin = len(tab)
5     if debut == (fin-1):
6         return tab[debut]
7     else:
8         milieu = (debut + fin)//2
9         gauche = mystere(tab, debut, milieu)
10        droite = mystere(tab, milieu, fin)
11        if (gauche > droite):
12            return gauche
13        else:
14            return droite
```

Soit la liste :

```
1 tab = [5, 71, 23, 45, 28, 89, 63, 39]
```

1. Dessiner l'arbre des séparations engendré par la fonction sur la liste *tab*.
2. Dessiner l'arbre des recombinaisons. Quelle valeur renvoie l'appel **mystere(tab)** ?
3. Que fait cette fonction ?