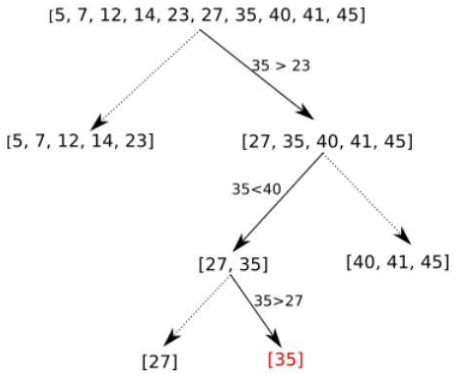
Diviser pour régner

1. La méthode diviser

La méthode diviser est le principe appliqué lors d'une recherche dichotomique dans une liste triée.
Dans l'exemple ci-contre le chiffre recherché est 35.
À chaque étape, on coupe l'intervalle de recherche en deux, et on en choisit une moitié en fonction du résultat de la comparaison entre l'élément recherché et la valeur médiane du tableau.
Comme les mêmes étapes se répètent, il est possible de le traiter de manière récursive : on teste si l'élément médian de la liste est égal à **x**, sinon on cherche sa présence à gauche ou à droite dans la liste suivant qu'il est plus grand ou plus petit que **x**.



✍ Compléter l'algorithme de la fonction récursive **recherche_dich**.

```
fonction recherche_dich (x, liste)
    si liste est vide alors
        retourner . . . . . .
    élément median ← . . . . . .
    si . . . . . = x
        retourner vrai
    sinon si . . . . . .
        retourner recherche_dich(x, . . . . . .)
    sinon
        retourner recherche_dich(x, . . . . . .)
```

✍ Implémenter la fonction récursive **recherche_dich**.

2. La méthode diviser pour régner

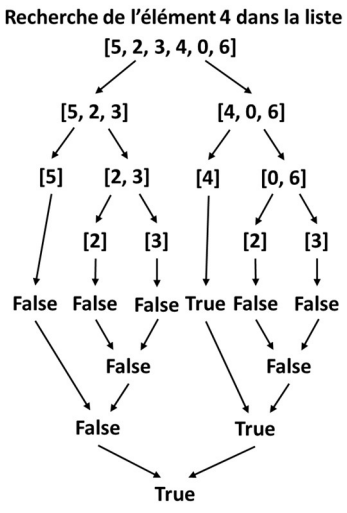
La méthode repose sur trois étapes :

- **Diviser** : le problème d'origine est divisé en un certain nombre de sous problème du même type.
- **Régner** : on résout les sous-problèmes (les sous problèmes sont plus faciles à résoudre que le problème d'origine).
- **Combiner** : les solutions des sous-problèmes sont combinées afin d'obtenir la solution du problème d'origine.

2.1. Recherche d'un élément dans une liste

Le but est de tester la présence d'un élément dans une liste contenant au moins un élément en utilisant la méthode diviser pour régner.
La recherche répond à la description suivante :

- Si la liste contient un seul élément et que cet élément est celui recherché alors retourner vrai sinon retourner faux.
- Sinon on divise la liste en deux sous listes par la moitié et on teste récursivement si l'élément est dans la première sous-liste ou dans la 2ième sous-liste.



✍ Compléter l'algorithme de la fonction récursive **recherche_rec**.

```
fonction recherche_rec(elt, liste)
    si . . . . . .
        si . . . . . .
            retourner . . . . . .
        sinon
            retourner . . . . . .
    sinon
        milieu ← . . . . . .
        retourner recherche_rec (elt, . . . . . .) ou recherche_rec (elt, . . . . . .)
```

✍ Réaliser un arbre comme ci-dessus en appliquant l'algorithme ci-dessus dans le cas de la recherche de l'élément **5** dans la liste **[4, 6, 0, 2]**.
✍ Sur l'arbre, repérer phases **diviser**, **régner** et **combiner**.

✍ Implémenter la fonction récursive **recherche_rec**.
✍ Compléter la fonction avec un **docstring** et des **doctest** significatifs.

2.1. Recherche du nombre d'occurrence d'un élément dans une liste

✍ À l'aide de la méthode diviser pour régner, implémenter une fonction **rech_nbre_occur** qui recherche le nombre d'occurrence d'un élément dans une liste.

Les assertions suivantes doivent être vérifiées par la fonction.

```
assert rech_nbre_occur(0, [1, 2, 5, 3, 2, 5]) == 0
assert rech_nbre_occur(2, [1, 2, 5, 3, 2, 5]) == 2
assert rech_nbre_occur("e", "element") == 3
assert rech_nbre_occur("e", ["element", 2, 3, "e"]) == 1
```