Résumé

Recherche dichotomique

La recherche dichotomique est un algorithmique qui répond à la question :

> Le tableau `tab`, **trié par ordre croissant** contient-il la `cle` ?

La **précondition** : *trié par ordre croissant* est indispensable sans quoi l'algorithmique ne peut être employé.

Coût

Le coût d'une recherche dichotomique est *logarithmique* en la taille du tableau.

Par exemple, pour un tableau à 100 éléments :

$64 < 100 < 128 \Leftrightarrow 2^6 < 100 < 2^7$

Donc une recherche dichotomique sur un tableau de taille 100 utilisera **au plus 7 tours de boucle.**.

Variantes

Des variantes existent, par exemple, plutôt que de renvoyer vrai ou faux on peut indiquer l'indice de l'élément en question.

La précondition et l'algorithmique restent les mêmes.

Et sans la précondition ?

Si le tableau n'est pas trié, alors c'est une recherche par balayage (un parcours séquentiel).

Le coût d'une recherche par balayage est linéaire : il faut au plus 100 tours de boucle pour un tableau de taille 100.

Est-ce utile de trier un tableau puis d'appliquer une recherche dichotomique ?

- Cela fonctionne,
- C'est inutile de le faire si on ne cherche qu'une seule fois,
- Cela peut devenir intéressant si on cherche très souvent.

Principe

C'est la **stratégie du "plus ou moins"** : viser le centre et éliminer la moitié des nombres.

À chaque tour, on vise le milieu des données.

On compare alors la clé avec la valeur située au milieu.

- Si la clé est égale, c'est trouvé.
- Si la clé est plus petit que le milieu, on recommence dans la partie de gauche (avant le milieu)
- Si la clé est plus grande, on recommence dans la partie droite (après le milieu)

Exemple à la main

`g` l'indice de "gauche", `d` celui de droite, `^` indique le milieu.

1. Premier tour

```
x = 5
T = [ 1,  3,  5,  7,  9, 11, 13, 15]
      g           ^              d
```

```
g = 0, d = 7
m = (g + d) // 2 = (0 + 7) // 2 = 3
    x = 5 < T[3] = 7    =>   Chercher à gauche
```

On recommence avec

- `d = m - 1 = 2`
- `g` inchangé

2. Second tour

```
x = 5
T = [ 1,  3,  5,  7,  9, 11, 13, 15]
      g   ^   d
```

```
g = 0, d = 2
m = (g + d) // 2 = (0 + 2) // 2 = 1
    x = 5 > T[1] = 3    =>   Chercher à droite
```

On recommence avec

- `d` inchangé
- `g = m + 1 = 2`

3. Troisième tour

```
x = 5
T = [ 1,  3,  5,  7,  9, 11, 13, 15]
             g=d
```

```
g = 2, d = 2
m = (g + d) // 2 = (2 + 2) // 2 = 2
    x = 5 == T[2] = 5    =>   Trouvé !
```

On peut renvoyer 2.

Langage naturel

Version qui renvoie **l'indice de la clé** ou -1 si la clé ne figure pas dans le tableau.

```
recherche_dichotomique(T: un tableau trié, x: la clé)
    g = 0, d = longueur du tableau - 1
    tant que g <= d,
        m = milieu de g et d
        si T[m] == x, renvoyer m
        si T[m] > x, g = m + 1
        si T[m] < x, d = m - 1

    Si la boucle se termine, renvoyer -1
```

Version Python

```python
def recherche_dichotomique(T: list, x: int) -> int:
    """
    Renvoie l'indice de `x` dans `T`.
    Renvoie -1 si `x` n'est pas dans `T`.

    Précondition : `T` est trié par ordre croissant
    """
    g = 0
    d = len(T) - 1
    while g <= d:
        m = (g + d) // 2
        if x == T[m]:
            return m
        elif x > T[m]:
            g = m + 1
        else:
            d = m - 1
    return -1
```

S'emploie comme ça :

```
>>> recherche_dichotomique([1, 3, 5, 7, 9], 2)
-1
>>> recherche_dichotomique([1, 3, 5, 7, 9], 7)
3
```