

Recherche dichotomique dans un tableau

Contexte

- ▶ Un tableau trié : $T = [0, 1, 2, \dots, 9]$
- ▶ L'élément 3 est-il dans le tableau ?
- ▶ L'objectif : répondre Oui ou Non en réalisant **le moins d'opérations** possibles.

Dichotomie

- ▶ À chaque étape on teste la valeur centrale
- ▶ Si c'est l'élément cherché, on a trouvé et la réponse est Oui.

Dichotomie

- ▶ À chaque étape on teste la valeur centrale
- ▶ Si c'est l'élément cherché, on a trouvé et la réponse est Oui.
- ▶ **Si la valeur centrale est supérieure à l'élément cherché on recommence avec la partie gauche**

Dichotomie

- ▶ À chaque étape on teste la valeur centrale
- ▶ Si c'est l'élément cherché, on a trouvé et la réponse est Oui.
- ▶ Si la valeur centrale est supérieure à l'élément cherché on recommence avec la partie gauche
- ▶ **Sinon on recommence avec la partie droite.**

Dichotomie

- ▶ À chaque étape on teste la valeur centrale
- ▶ Si c'est l'élément cherché, on a trouvé et la réponse est Oui.
- ▶ Si la valeur centrale est supérieure à l'élément cherché on recommence avec la partie gauche
- ▶ Sinon on recommence avec la partie droite.
- ▶ **Si la partie gauche ou la partie droite est vide, l'élément n'est pas dans le tableau et la réponse est : Non.**

Déroulé sur l'exemple, à la main.

$T = [0, 1, 2, \dots, 9]$. On cherche 3.

1. On propose : 4.

$4 > 3$ donc *on recommence avec la partie **avant 5** : $T1 = [0, 1, 2, 3]$*

Déroulé sur l'exemple, à la main.

$T = [0, 1, 2, \dots, 9]$. On cherche 3.

1. On propose : 4.

$4 > 3$ donc on recommence avec la partie avant 5 : $T_1 = [0, 1, 2, 3]$

2. On propose : 2

$2 < 3$ donc *on recommence avec la partie **après** 2* : $T_2 = [3]$

Déroulé sur l'exemple, à la main.

$T = [0, 1, 2, \dots, 9]$. On cherche 3.

1. On propose : 4.

$4 > 3$ donc on recommence avec la partie avant 5 : $T_1 = [0, 1, 2, 3]$

2. On propose : 2

$2 < 3$ donc on recommence avec la partie après 2 : $T_2 = [3]$

3. On propose : 3

$3 = 3$ donc **on a trouvé l'élément et la réponse est : Oui, 3 est dans T.**

L'algorithme

```
rechercheDicho(liste, clé)
    bas = 0
    haut = longueur(liste) - 1
    Tant que (bas < haut) :
        med = (bas + haut) // 2
        si clé == liste[med]:
            bas = med ;
            haut = med
        Sinon :
            si clé > liste[med] {bas = med + 1}
            sinon: haut = med - 1
    si cle == liste[bas]: renvoyer Vrai
    sinon: renvoyer Faux
```

Même exemple, avec les variables

Voici un déroulé de l'algorithme à la main.

Notre tableau T est [0, 1, 2, ..., 9] et on cherche 3.

On dispose des variables :

- ▶ début, milieu, fin qui sont des éléments du tableau
- ▶ trouvé qui est un booléen (vrai / faux)
- ▶ et `val > milieu` (booléen, qui va nous aider à choisir)

On présente les éléments dans une table :

Même exemple, avec les variables

Voici un déroulé de l'algorithme à la main.

Notre tableau T est [0, 1, 2, ..., 9] et on cherche 3.

- **Avant la boucle.** début, fin et trouvé sont initialisées (0, 9, faux). La variable milieu et le booléen `val > milieu` n'existent pas encore.

Tour	début	milieu	fin	trouvé	<code>val > milieu</code>
Avant la boucle	0	/	9	faux	/

Même exemple, avec les variables

Notre tableau T est [0, 1, 2, ..., 9] et on cherche 3.

1. **Premier tour.** On descend début et fin.

On calcule milieu $(0+9)/2 = 4.5$ dont la partie entière est 4.
Donc milieu = 4.

Est-ce que $3==4$? Faux.

Est-ce-que " $3>4$ " ? Faux. Dans ce cas, c'est fin qui prend la valeur de milieu

Tour	début	milieu	fin	trouvé	val > milieu
Avant la boucle	0	/	9	faux	/
1er tour	0	4	9	faux	$3 > 4$: faux

Même exemple, avec les variables

Notre tableau T est [0, 1, 2, ..., 9] et on cherche 3.

2. **Second tour.** On a descendu début, on donne à fin la valeur précédente de milieu (4). Et on calcule les nouveaux éléments.
 $\text{milieu} = (0+4)/2 = 2$ (entier).

Est-ce-que $3==2$? Faux.

Est-ce-que $3>2$? Vrai. Dans ce cas, c'est début qui change et prend la valeur de milieu + 1.

Tour	début	milieu	fin	trouvé	val > milieu
Avant la boucle	0	/	9	faux	/
1er tour	0	4	9	faux	$3 > 4$: faux
2ème tour	0	2	4	faux	$3 > 2$: vrai

Même exemple, avec les variables

Notre tableau T est [0, 1, 2, ..., 9] et on cherche 3.

3. **Troisième tour.** On a descendu la valeur de fin et donné à début l'ancienne valeur de milieu + 1 (3). Et on recommence.

Milieu = $(3 + 4)/2 = 3.5$ dont la partie entière est 3.

Est-ce-que $3 == 3$? VRAI !!!

Tour	début	milieu	fin	trouvé	val > milieu
Avant la boucle	0	/	9	faux	/
1er tour	0	4	9	faux	$3 > 4$: faux
2ème tour	0	2	4	faux	$3 > 2$: vrai
3ème tour	3	3	4	vrai	/

L'algorithme est terminé et la sortie est "VRAI". Le nombre 3 est bien un élément du tableau [0, 1, **3**, 4, 5, ..., 9].

Conclusion

- ▶ La recherche dichotomique permet de gagner beaucoup d'étape par rapport au parcours séquentiel du tableau.
- ▶ Elle nécessite d'avoir un tableau **trié** sans quoi on ne peut l'appliquer.

Remarques sur la complexité

- ▶ Si on ne souhaite l'appliquer qu'une seule fois, il n'est pas toujours intéressant de trier de le tableau pour chercher. C'est généralement trop long...
- ▶ Mais si on doit souvent effectuer des recherches dans le tableau, alors c'est indispensable.

Nombre d'étapes

- ▶ **parcours séquentiel** : autant que d'éléments dans le tableau dans le pire des cas. Le parcours séquentiel prend (dans le pire des cas) n étapes.
- ▶ **recherche dichotomique** (après le tri) : $\log_2 n$ étapes. $\log_2 n$ est (grosso modo) le nombre de divisions entières de n par 2 qu'on peut effectuer avant de trouver un quotient nul.