

1. Contexte

Parcourir un tableau de manière linéaire est efficace s'il est de petite taille. Cette méthode fonctionne aussi pour des structures non ordonnées. La méthode par dichotomie permet de déterminer l'existence d'un élément de manière plus rapide sur des tableaux ordonnés de grande taille.

Principe de la dichotomie : Le tableau d'éléments est divisé en deux parties. L'élément au milieu du tableau est comparé à l'élément recherché. S'ils sont égaux, l'élément recherché est présent, sinon la partie contenant l'élément recherché est identifiée. Elle est à son tour divisée en deux parties. Le processus se poursuit jusqu'à obtenir un tableau d'un seul élément qui peut être l'élément recherché. Si le tableau final est vide, cela signifie que l'élément recherché n'est pas présent.

2. Algorithme de recherche par dichotomie

Un algorithme possible est le suivant.

Algorithme : Recherche dichotomique

Entrées : t : un tableau *trié* de n valeurs, x : un élément
Sorties : Vrai si l'élément x est dans t , Faux sinon

début

```

1   $debut \leftarrow 0$ 
2   $fin \leftarrow |t| - 1$ 
3   $trouve \leftarrow Faux$ 
4  tant que  $trouve = Faux$  et  $debut \leq fin$  faire
5       $m = \lfloor \frac{(debut+fin)}{2} \rfloor$ 
6      si  $x = t[m]$  alors
7           $trouve \leftarrow Vrai$ 
8      sinon si  $x > t[m]$  alors
9           $debut \leftarrow m + 1$ 
10     sinon
11          $fin \leftarrow m - 1$ 
12     fin
13 renvoyer  $trouve$ 
14 fin


```

 **Question 1** : Exécuter à la main l'algorithme en considérant le tableau t suivant et $x = 65$:

$t = [1, 1, 3, 10, 16, 18, 19, 22, 35, 41, 46, 52, 55, 59, 60, 65, 67, 80, 91, 100]$

Pour cela, compléter le tableau de valeurs au fur et à mesure de l'exécution de l'algorithme.

$debut$	fin	m	$t[m]$
0	19	9	

 **Question 2** : Combien d'itérations ont-été nécessaires pour trouver x ?

 **Question 3** : Combien d'itérations sont nécessaires avec l'algorithme séquentiel ?

 **Question 4** : Montrer que l'algorithme par dichotomie se termine.

3. Éléments sur la complexité temporelle

 Question 5 : Formuler le meilleur et le pire des cas pour l'algorithme de recherche par dichotomie.


 Question 6 : Voici des exemples d'exécutions de l'algorithme de recherche sur différents tableaux.

tableau t	x	I (nombre d'itérations de la boucle)	n (taille de t)	n_2 (représentation binaire de n)
[4, 9]	2	2		
[1, 5, 9, 13]	15	3		
[-5, -2, 0, 3, 7, 9, 14, 16]	-7	4		
[1, 4, 5, 6, 8, 9, 13, 14, 18, 23, 25]	-1	4		

1. Compléter les colonnes " n " et " n_2 " du tableau ci-dessus.
2. Quel lien est-il possible d'établir entre la taille p de n_2 et I le nombre d'itérations de la boucle de l'algorithme de recherche par dichotomie ?
3. Exprimer la complexité de l'algorithme de recherche par dichotomie en fonction de la taille n du tableau dans le pire des cas.

4. Implémentation en Python


 Question 7 : Implémenter en python l'algorithme de recherche par dichotomie dans un script `dichotomie.py`.

 Question 8 : Exécuter l'algorithme pour différentes valeurs de n . On prendra $n = 1, n = 2, n = 4, \dots$, jusque $n = 128$ (valeur de n doublée à chaque fois). On travaillera dans le pire des cas. Ajouter un compteur d'itérations et noter les valeurs obtenues.

Aide : La génération d'un tableau de valeurs aléatoires peut s'écrire via l'instruction

`t = [random.randint(0, 100) for i in range(30)]` grâce au module `random`.

Le tableau peut être triée avec l'instruction `t.sort()`.

 Question 9 : Tracer les nuages de points des compteurs d'itérations et de la fonction $f(n) = n$. Quel type de courbe obtient-on ? Est-ce attendu ? Expliquer.