

Problématique

Recherche  
classique dans un  
tableau

Génération des données  
Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées  
Recherche dichotomique  
Efficacité

# Recherche dichotomique

Christophe Viroulaud

Première - NSI

## 1. Problématique

## 2. Recherche classique dans un tableau

## 3. Recherche dans un tableau trié

### Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

Rechercher un élément dans un tableau est une opération courante. Cette tâche a un coût qui dépend de la taille du tableau. Cependant, si le tableau est déjà trié est-il possible d'accélérer la recherche ?

Comment implémenter une recherche efficace dans un tableau trié ?

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

## 1. Problématique

## 2. Recherche classique dans un tableau

### 2.1 Génération des données

### 2.2 Recherche dans les données

## 3. Recherche dans un tableau trié

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

Imaginons un supermarché qui référence chaque article par un entier. Les références, au nombre de cent mille, sont contenues dans un tableau.



12



6780



376



134900

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

**Activité 1 :** Construire par compréhension un tableau de cent mille entiers compris entre 0 et 1000000.

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

```
1 entiers = [randint(0, 1000000) for _ in range(100000)]
```

Jeu de données

1. Problématique
2. Recherche classique dans un tableau
  - 2.1 Génération des données
  - 2.2 Recherche dans les données
3. Recherche dans un tableau trié

Problématique

Recherche  
classique dans un  
tableau

Génération des données

**Recherche dans les données**

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité



Pour vérifier la présence d'une valeur dans les données, il faut parcourir le tableau élément par élément.

3	180	1007	56			2178	8
---	-----	------	----	--	--	------	---

FIGURE 1 – Parcours séquentiel

# Dans le pire des cas

Problématique

Recherche  
classique dans un  
tableau

Génération des données

**Recherche dans les données**

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

nombre d'éléments	nombre de comparaisons
100	100
10000	10000
1000000	1000000

## À retenir

Dans le pire des cas le nombre d'opérations de la recherche dépend du nombre d'éléments.

La complexité est **linéaire**.

## Activité 2 :

1. Écrire la fonction `recherche_classique(tab: list, cherche: int) → bool` qui renvoie `True` si l'entier `cherche` est présent dans le tableau.
2. Tester la fonction : vérifier si le nombre 575000 a été choisi par une personne.

Problématique

Recherche  
classique dans un  
tableau

Génération des données

**Recherche dans les données**

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

```
1  """
2  Renvoie True si 'cherche' est dans 'tab'
3  """
4  for element in tab:
5      if element == cherche:
6          return True
7  # à la fin de la boucle on n'a pas trouvé 'cherche'
8  return False
```

## Activité 2 :

3. Dans le programme principal, créer une variable **COMPTEUR** initialisée à 0. Cette variable de test sera utilisée *dans la fonction* pour compter le nombre d'itérations de la boucle de recherche. On parle alors de **variable globale** car elle n'est pas propre à la fonction. Il faudra ajouter le code 1 au début de la fonction.

```
1 global COMPTEUR
```

Code 1 – Déclaration d'une variable globale

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

```
1 COMPTEUR = 0
2
3 def recherche_classique(tab: list, cherche: int) -> bool:
4     """
5     Renvoie True si 'cherche' est dans 'tab'
6     """
7     global COMPTEUR
8     for element in tab:
9         COMPTEUR += 1
10        if element == cherche:
11            return True
12    # à la fin de la boucle on n'a pas trouvé 'cherche'
13    return False
```

1. Problématique
2. Recherche classique dans un tableau
3. Recherche dans un tableau trié
  - 3.1 Des données ordonnées
  - 3.2 Recherche dichotomique
  - 3.3 Efficacité

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité



Considérons maintenant que les références sont triées par ordre croissant au fur et à mesure de leur ajout dans le tableau de données.

alimentaire				vêtement			électro-ménager	
3	8	56	180		1007	2178	8000	1160012130

FIGURE 2 – Références triées

**Activité 3 :** Pour simplifier nous allons utiliser la méthode `sort` pour trier les données.

1. Construire par compréhension un tableau de cent mille entiers compris entre 0 et 1000000.
2. Trier le tableau.

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

```
1 entiers = [randint(0, 1000000) for _ in range(100000)]  
2 entiers.sort()
```

Jeu de données

## 1. Problématique

## 2. Recherche classique dans un tableau

## 3. Recherche dans un tableau trié

### 3.1 Des données ordonnées

### 3.2 Recherche dichotomique

### 3.3 Efficacité

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

**Recherche dichotomique**

Efficacité

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

Les données étant triées, le principe de la dichotomie, pour chercher la présence d'un élément, consiste à :

- ▶ couper le tableau en deux parties égales,
- ▶ ne garder que la partie contenant l'élément,
- ▶ répéter l'opération jusqu'à trouver l'élément ou avoir une partie vide.

Cherchons 302 dans le tableau suivant :

3	8	56	180	256	302	765	1007	2178
---	---	----	-----	-----	-----	-----	------	------

FIGURE 3 – Séparons les données en deux parties

3	8	56	180	256	302	765	1007	2178
---	---	----	-----	-----	-----	-----	------	------

FIGURE 4 – 256 n'est pas le nombre recherché et il est inférieur à 302

3	8	56	180	256	302	765	1007	2178
---	---	----	-----	-----	-----	-----	------	------

FIGURE 5 – Séparons les données restantes en deux parties



3	8	56	180	256	302	765	1007	2178
---	---	----	-----	-----	-----	-----	------	------

FIGURE 6 – Nous pouvons éliminer la partie supérieure.

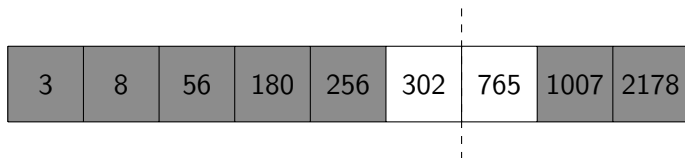


FIGURE 7 – Dernière séparation

3	8	56	180	256	302	765	1007	2178
---	---	----	-----	-----	-----	-----	------	------

FIGURE 8 – 302 a été trouvée en trois itérations

En pratique, on utilise les indices pour trouver le milieu.

0	1	2	3	4	5	6	7	8
3	8	56	180	256	302	765	1007	2178

FIGURE 9 –  $\frac{8+0}{2} = 4$  l'indice médian est 4

```
1 i_debut = 0
2 i_fin = 8
```

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

0	1	2	3	4	5	6	7	8
3	8	56	180	256	302	765	1007	2178

FIGURE 10 – 256 n'est pas le nombre recherché

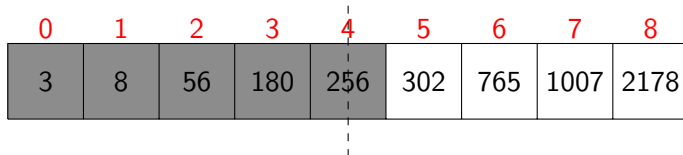


FIGURE 11 – 256 est inférieur au nombre recherché.

```

1 i_debut = 5
2 i_fin = 8

```

Problématique

Recherche  
classique dans un  
tableau

Génération des données  
Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées  
**Recherche dichotomique**  
Efficacité

0	1	2	3	4	5	6	7	8
3	8	56	180	256	302	765	1007	2178

FIGURE 12 –  $\frac{8 + 5}{2} = 6$  l'indice médian est 6

Problématique

Recherche  
classique dans un  
tableau

Génération des données  
Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées  
**Recherche dichotomique**  
Efficacité

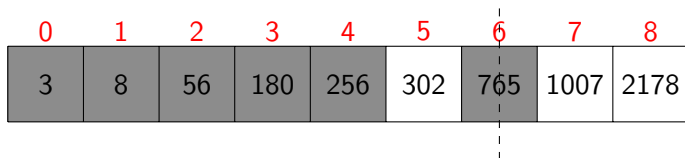


FIGURE 13 – 765 n'est pas le nombre recherché.



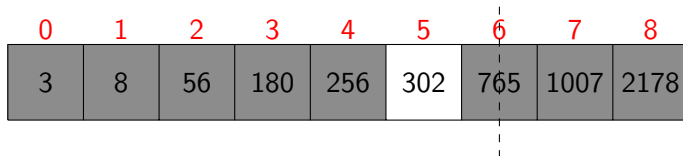


FIGURE 14 – 765 est supérieur au nombre recherché.

```

1 i_debut = 5
2 i_fin = 5

```

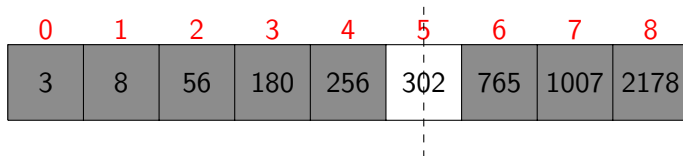


FIGURE 15 –  $\frac{5 + 5}{2} = 5$  l'indice médian est 5.

Problématique

Recherche  
classique dans un  
tableau

Génération des données  
Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées  
**Recherche dichotomique**  
Efficacité

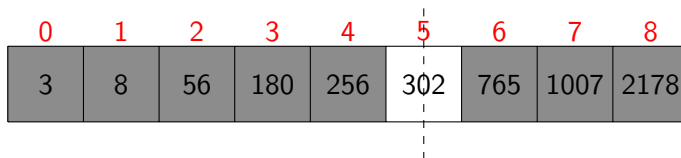


FIGURE 16 – On a trouvé l'élément.

**Activité 4 :** Écrire la fonction `recherche_dicho(tab: list, cherche: int) → bool` qui applique le principe de la dichotomie. Pour séparer les données en deux parties (à peu près) égales il faudra calculer l'indice médian de la partie encore valide.

# Correction

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

**Recherche dichotomique**

Efficacité

0	1	2	3	4	5	6	7	8
3	8	56	180	256	302	765	1007	2178

```
1 i_debut = 0
2 i_fin = len(tab)-1
3 while i_fin >= i_debut:
```

# Correction

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

**Recherche dichotomique**

Efficacité

0	1	2	3	4	5	6	7	8
3	8	56	180	256	302	765	1007	2178

```
1 i_milieu = (i_debut+i_fin) // 2
2 if cherche == tab[i_milieu] :
```

# Correction

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

**Recherche dichotomique**

Efficacité

0	1	2	3	4	5	6	7	8
3	8	56	180	256	302	765	1007	2178

```
1     return True
2     elif cherche < tab[i_milieu] :
```

# Correction

## Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

0	1	2	3	4	5	6	7	8
3	8	56	180	256	302	765	1007	2178

```
1     i_fin = i_milieu-1
2     else: # cherche > tab[i_milieu]
3         i_debut = i_milieu+1
4     # à la fin de la boucle on n'a pas trouvé 'cherche'
```



Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

```
1 i_debut = 0
2 i_fin = len(tab)-1
3 while i_fin >= i_debut:
4     i_milieu = (i_debut+i_fin) // 2
5     if cherche == tab[i_milieu] :
6         return True
7     elif cherche < tab[i_milieu] :
8         i_fin = i_milieu-1
9     else: # cherche > tab[i_milieu]
10         i_debut = i_milieu+1
11 # à la fin de la boucle on n'a pas trouvé 'cherche'
12 return False
```

1. Problématique
2. Recherche classique dans un tableau
3. Recherche dans un tableau trié
  - 3.1 Des données ordonnées
  - 3.2 Recherche dichotomique
  - 3.3 Efficacité

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

## Activité 5 :

1. En utilisant une variable **COMPTEUR**, compter le nombre d'itérations de la boucle de recherche dichotomique.
2. Tester pour différentes tailles de tableau.

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

```
1 COMPTEUR = 0
2
3 def recherche_dicho(tab: list, cherche: int) -> bool:
4     global COMPTEUR
5     i_debut = 0
6     i_fin = len(tab)-1
7     while i_fin >= i_debut:
8         COMPTEUR += 1
9         i_milieu = (i_debut+i_fin) // 2
10        if cherche == tab[i_milieu] :
11            return True
12        elif cherche < tab[i_milieu] :
13            i_fin = i_milieu-1
14        else: # cherche > tab[i_milieu]
15            i_debut = i_milieu+1
16        # à la fin de la boucle on n'a pas trouvé 'cherche'
17    return False
```

À chaque itération la quantité de données (notée ***n***) à étudier est divisée par deux. Dans le pire des cas, on divise jusqu'à ce que la taille de la partie restante soit inférieure ou égale à 1.

$$\frac{n}{2^x} = 1$$
$$\Leftrightarrow n = 2^x$$

## Activité 6 :

1. Encadrer la valeur de  $x$  entre deux entiers, si le tableau contient  $n = 10000$  éléments.
2. Effectuer le même encadrement pour cent mille, un million d'éléments.

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

$$2^{13} = 8192 < x < 2^{14} = 16384$$

# Dans le pire des cas

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

nombre d'éléments	nombre de comparaisons
10	3-4
100	6-7
1000	9-10
10000	13-14
100000	16-17
1000000	19-20



## À retenir

La complexité temporelle de la recherche dichotomique est **logarithmique** :

$$\log_2 n = x$$

# Code complet

Recherche  
dichotomique

Problématique

Recherche  
classique dans un  
tableau

Génération des données

Recherche dans les données

Recherche dans un  
tableau trié

Des données ordonnées

Recherche dichotomique

Efficacité

Le code complet se trouve [ici](#).