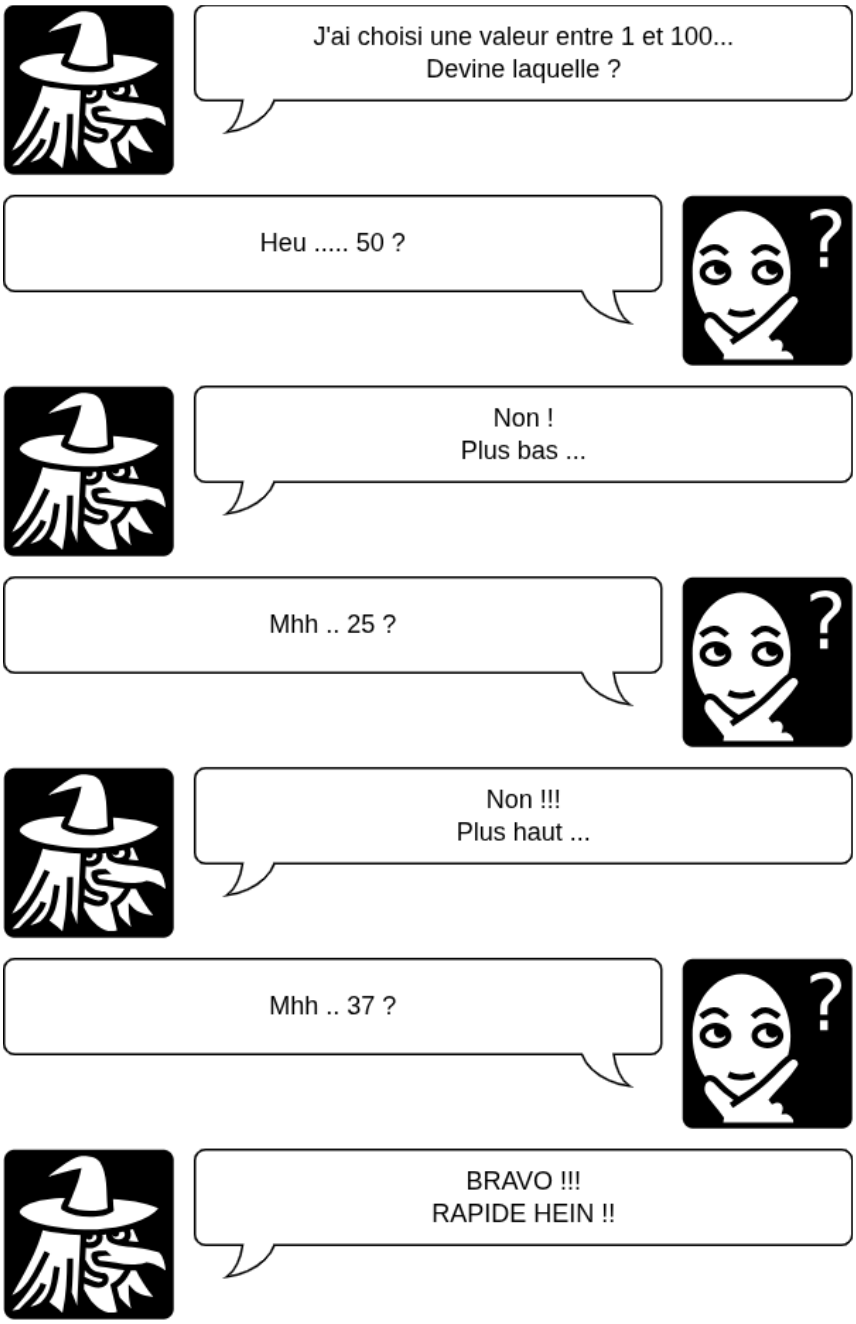
Nom :	Prénom :	Classe :

NSI 1re — Algorithme de recherche dichotomique (dans un tableau trié)

Objectifs :
- Savoir reconnaître un algorithme de recherche dichotomique.
- Savoir expliquer son fonctionnement.
- Savoir écrire un algorithme de recherche dichotomique dans un tableau indexé trié.

Introduction

Essayez de décrire comment notre camarade fait pour trouver *si rapidement* la valeur choisie par la sorcière ?



La recherche dichotomique

Un algorithme de *recherche dichotomique*[1] est une méthode qui permet de **trouver rapidement la position** d'un élément dans un tableau déjà trié en **divisant sans cesse l'intervalle de recherche en deux**.

C'est comme chercher un mot dans un dictionnaire papier : on n'ouvre pas à la première page pour tout parcourir, on ouvre vers le milieu, on regarde si on est avant ou après le mot voulu, puis on resserre la recherche dans la bonne moitié, et ainsi de suite.

Le fait que l'on travaille sur un tableau déjà trié (par exemple par ordre croissant) facilite grandement notre opération de recherche d'un élément. En effet, il suffit de comparer la **valeur recherchée** avec la **valeur située au milieu** du tableau.

- Si la valeur recherchée est **plus grande** que celle du milieu, on peut restreindre la recherche à la **moitié droite** du tableau.
- Si la valeur recherchée est **plus petite** que celle du milieu, on peut restreindre la recherche à la **moitié gauche** du tableau.

En répétant ce procédé, on divise la zone de recherche par deux à chaque fois.

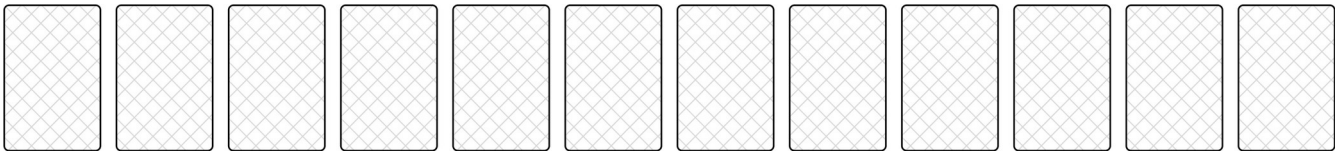
Très rapidement, on parviendra soit à la valeur recherchée, soit à un intervalle devenu vide.

On appelle cela la recherche dichotomique. On connaît tous ce principe, pour l'avoir appliqué étant enfant pour deviner un nombre en ayant comme réponses à nos tentatives « c'est plus petit » ou « c'est plus grand ». *Et c'est aussi ce qui est illustré avec la sorcière page précédente.*

Ce principe est connu en informatique sous le nom de « diviser pour régner[2] » et il est appliqué dans de nombreux algorithmes. La recherche dichotomique en est l'expression la plus simple.

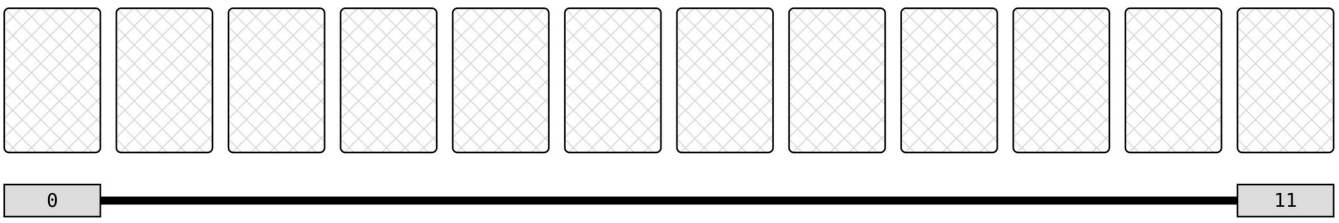
Déroulé visuel

Prenons un exemple dans lequel nous avons 12 cartes devant nous, triées dans l'ordre croissant :



Imaginons que nous soyons à la recherche de la carte valant `9` ...

Cette **carte recherchée** peut être *n'importe où* entre la carte d'indice `0` et la carte d'indice `11` :



1 Dichotomie vient du grec ancien : díkha « en deux, séparé » et tomós « coupure ». Donc « coupure en deux ».
2 Le principe « diviser pour régner » sera vu en terminale.

Pour regarder la carte « du milieu », il nous faut définir la position (l'indice) du milieu.

Pour trouver cet **indice du milieu**, nous pouvons faire le calcul suivant :

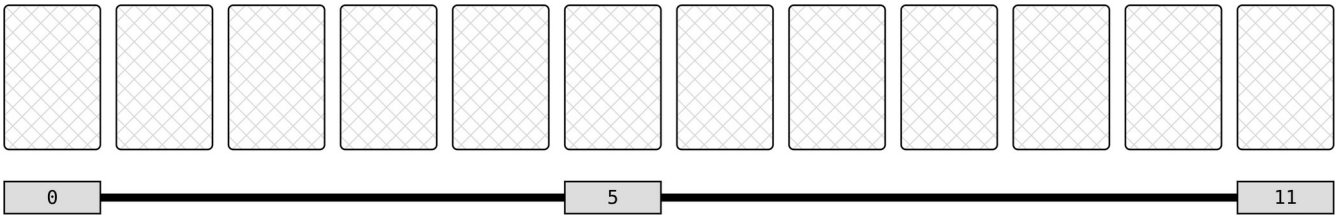
```
( indice de gauche + indice de droite ) // 2
```

Pour rappel, l'opérateur `//` permet d'obtenir le quotient (le résultat d'une division).

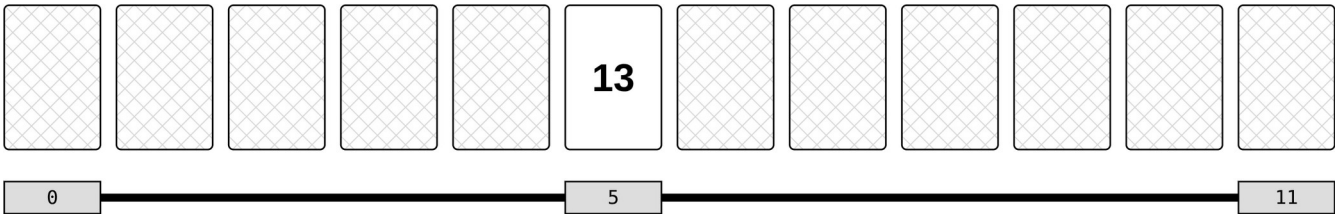
Essayons de calculer l'indice du milieu actuel :

```
( 0 + 11 ) // 2
```

Donc notre indice du milieu vaut `(0 + 11) // 2` soit `5` :



Regardons la valeur de cette carte d'indice 5 :



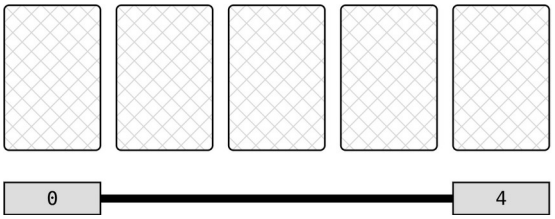
Pour rappel, la **carte recherchée** est la carte de valeur 9.
- Est-ce que cette **carte du milieu** vaut 9 ? Non.
- Est-ce que la **carte recherchée** est plus petite que cette **carte du milieu** ? Oui.

- Si la carte recherchée est plus petite que la valeur du milieu, on doit faire une nouvelle recherche à **gauche**.
- Si la carte recherchée est plus grande que la valeur du milieu, on doit faire une nouvelle recherche à **droite**.
- Si la carte recherchée est égale à la valeur du milieu, on a gagné !

Dans notre cas, nous devons faire une recherche à gauche.

Faire une recherche à gauche :	**Faire une recherche à droite** :
- L'indice de gauche reste inchangé. - L'indice de droite vaut (indice du milieu - 1)	- L'indice de droite reste inchangé. - L'indice de gauche vaut (indice du milieu + 1)

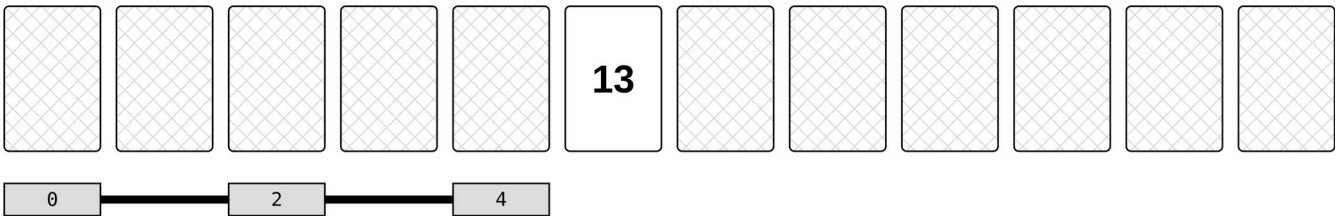
Dans notre cas, l'indice de gauche reste inchangé, et l'indice de droite vaut *indice du milieu - 1*, donc `4` :



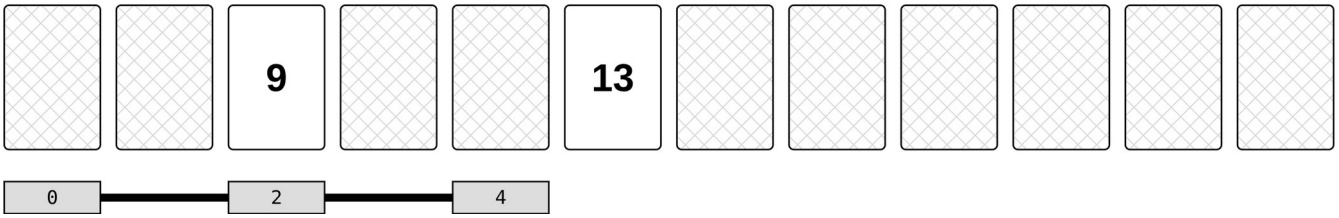
Essayons de calculer notre nouvel **indice du milieu**, *donc en faisant (indice gauche + indice droite) // 2* :

```
( 0 + 4 ) // 2
```

Donc notre indice du milieu vaut `(0 + 4) // 2` soit `2` :



Regardons la valeur de cette carte du milieu d'indice 2 :



Bingo, c'est la carte que nous cherchions !

Déroulé compact

Il est possible de représenter cette méthode de recherche dichotomique sous la forme d'un *déroulé compact*. Imaginons que nous ayons cinq éléments (triés) devant nous, et que nous soyons à la recherche de la valeur **13** :

Carte	Carte	Carte	Carte	Carte	i_gauche	i_droite	i_milieu
1	7	12	13	15			
1	7	12	13	15	0	4	2
1	7	12	13	15	3	4	3

- Sur la première ligne, on a bien cinq éléments (cinq « cartes »).

- Sur la deuxième ligne :
  - On définit un indice de gauche `i_gauche` à `0`, *c'est l'indice du premier élément.*
  - On définit un indice de droite `i_droite` à `4`, *c'est l'indice du dernier élément.*
  - On définit un indice de milieu `i_milieu` à `2`, *calculé via* `(i_gauche + i_droite) // 2`
  - Puis on se pose la question : est-ce que la valeur cherchée est inférieure, supérieure ou égale à celle du milieu ?
    - Ici nous cherchons la valeur 13, et 13 est supérieure à la valeur du milieu (12)…
      Donc il nous faut définir un nouvel indice de gauche…

- Sur la troisième ligne :
  - On définit un indice de gauche `i_gauche` à `3`, *calculé via le précédent* `i_milieu + 1`.
  - On garde le même indice de droite.
  - On définit un indice de milieu `i_milieu` à `3`, *calculé via* `(i_gauche + i_droite) // 2`
  - Puis on se pose la question : est-ce que la valeur cherchée est inférieure, supérieure ou égale à celle du milieu ?
    - Ici nous cherchons la valeur 13, et 13 est égale à la valeur du milieu (13)…

Donc la valeur cherchée (13) est bien présente dans le tableau, elle est située à l'indice 3.

➤ **Exercice 2** — Enquêter

a) Nous cherchons la valeur 50 dans le tableau trié suivant. Complétez-en le *déroulé compact* :

c	c	c	c	c	c	c	c	c	c	i_gauche	i_droite	i_milieu
1	3	4	7	8	9	20	21	50	97			
1	3	4	7	8	9	20	21	50	97	0	9	4
1	3	4	7	8	9	20	21	50	97	5	9	7
1	3	4	7	8	9	20	21	50	97	8	9	8

b) Nous cherchons la valeur 43 dans le tableau trié suivant. Complétez-en le *déroulé compact* :

c	c	c	c	c	c	c	c	c	c	i_gauche	i_droite	i_milieu
7	8	42	43	51	60	91	92	95	96			
7	8	42	43	51	60	91	92	95	96	0	9	4
7	8	42	43	51	60	91	92	95	96	0	3	1
7	8	42	43	51	60	91	92	95	96	2	3	2
7	8	42	43	51	60	91	92	95	96	3	3	3

Implémentation en pseudo-code

Comme tout algorithme, la méthode de la recherche dichotomique d'un tableau trié peut être implémentée dans n'importe quel langage de programmation. Une première approche pourrait être d'analyser une implémentation possible en *pseudo-code* :

```
1   i_gauche = 0
2   i_droite = Taille(tableau) - 1
3
4   Tant que i_gauche ≤ i_droite
5       i_milieu = (i_gauche + i_droite) // 2  # Calcul de l'indice de milieu
6       v_milieu = tableau[i_milieu]           # Récupération de sa valeur
7
8       Si v_cherchée < v_milieu               # La valeur cherchée est plus petite
9           i_droite = i_milieu - 1            # Resserre l'indice de droite
10      Sinon si v_cherchée > v_milieu         # La valeur cherchée est plus grande
11          i_gauche = i_milieu + 1            # Resserre l'indice de gauche
12      Sinon
13          Afficher("La valeur cherchée est à l'indice", i_milieu)
14          Arrêter la boucle
```

Implémentation en Python

➤ Exercice 3 — Enquêter

Convertissez le pseudo-code de la recherche dichotomique d'un tableau trié en langage Python :

```
i_gauche = 0
i_droite = len(tableau) - 1

# Début du parcours
while i_gauche <= i_droite:
    i_milieu = (i_gauche + i_droite) // 2 # Calcul de l'indice de milieu
    v_milieu = tableau[i_milieu] # Récupération de sa valeur

    if v_cherchee < v_milieu:
        # La valeur cherchée est plus petite
        i_droite = i_milieu - 1 # Resserre l'indice de droite
    elif v_cherchee > v_milieu:
        # La valeur cherchée est plus grande
        i_gauche = i_milieu + 1 # Resserre l'indice de gauche
    else:
        # C'est la valeur cherchée !
        print("La valeur cherchée est à l'indice", i_milieu)
        break
```

Pour aller plus loin

- Limites techniques et risques : La recherche dichotomique suppose absolument que les données soient triées selon le même critère que celui utilisé pour la comparaison ; si ce n’est pas le cas (données mal triées, doublons mal gérés, erreurs de tri), le résultat peut être faux tout en paraissant « raisonnable ». Dans des systèmes critiques (finance, transport, médical), une recherche incorrecte dans une base mal ordonnée peut conduire à sélectionner un mauvais dossier, un mauvais patient ou un mauvais paramètre, avec des conséquences potentiellement graves.

- Limites conceptuelles : On retrouve aussi cette logique de « couper en deux » dans des procédures historiques comme certains contrôles, enquêtes ou filtrages administratifs où l’on cherche à réduire rapidement une liste de suspects ou de dossiers, ce qui peut être efficace mais aussi injuste si l’on se fonde sur des indices mal choisis. Des controverses récentes autour d’algorithmes de tri d’élèves ou de candidats (admissions, bourses, aides sociales) ont précisément mis en lumière le risque de prendre des décisions automatiques rapides sur la base de données mal préparées ou mal ordonnées, ce qui rejoint la limite fondamentale de la recherche dichotomique : elle ne corrige pas les erreurs de tri ou de modèle, elle ne fait que les exploiter plus vite.