



N.S.I

Algorithme de recherche dichotomique dans un
tableau trié

Sommaire

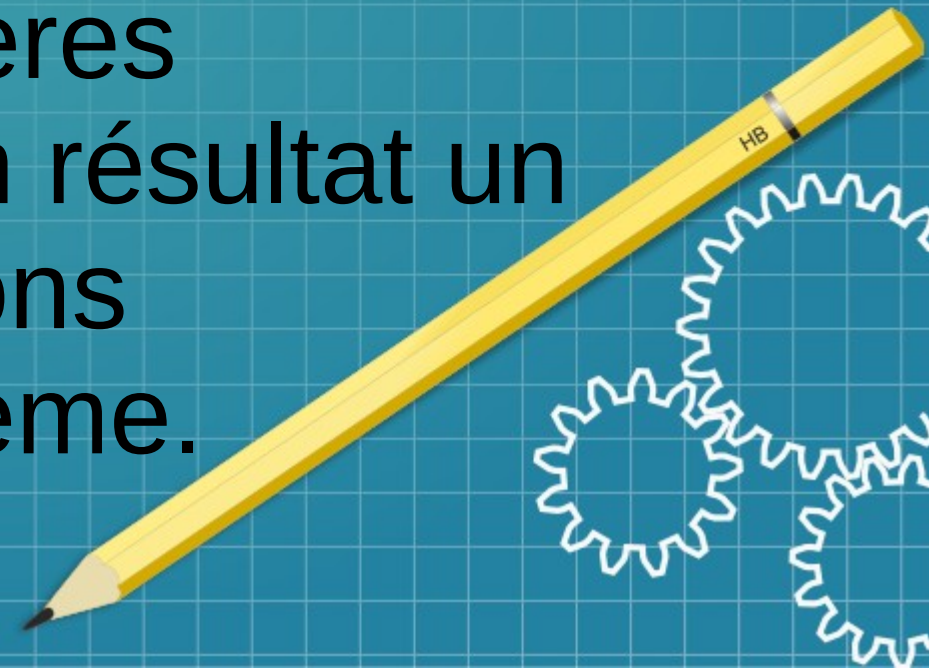


I) C'est quoi un algorithme ?	page 3
II) Objectif de notre algo	page 5
III) Principe de notre algo	page 6
IV) Où utiliser l'algo ?	page 8
V) Exemple d'algo d'une recherche dichotomique simple	page 11
VI) Variant de boucle	page 12

I) C'est quoi un algorithme



En informatique, un algorithme de recherche est un type d'algorithme qui, pour un domaine, un problème de ce domaine et des critères donnés, retourne en résultat un ensemble de solutions répondant au problème.





Problème

ALGO

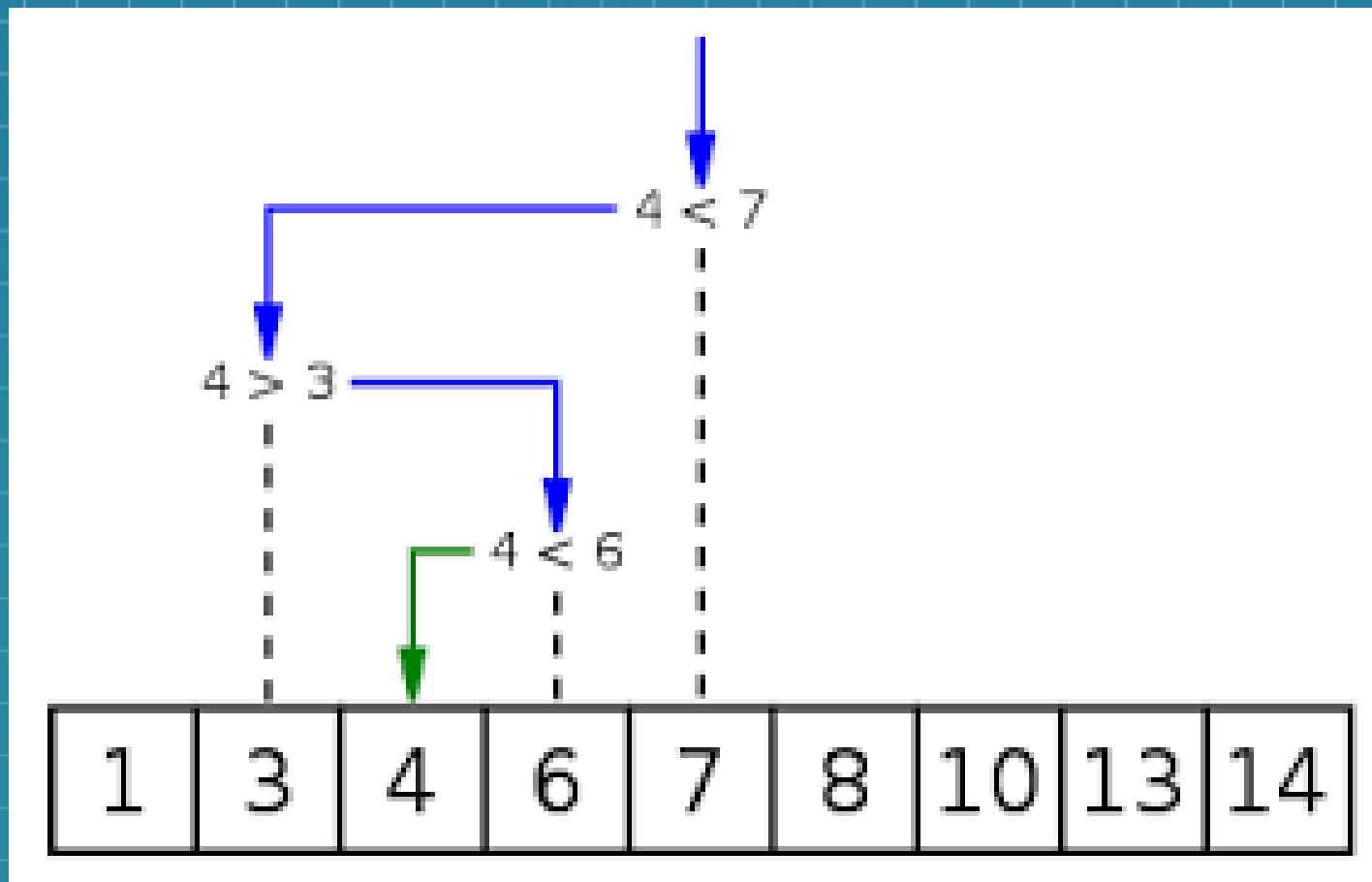
Solution

II) Objectif de notre algo

- La recherche dichotomique est un algorithme de recherche qui permet de trouver la position d'un élément dans un tableau trié.

III) Principe de notre algo

- La recherche dichotomique s'applique à des listes triées.
- Elle consiste tout d'abord à comparer l'élément recherché avec celui du milieu de la liste.
- Si il lui est égal, on arrête la recherche.
- Sinon soit l'élément recherché est plus petit que l'élément du milieu et on continue la recherche dans la première partie du tableau. Soit l'élément recherché est plus grand que l'élément du milieu et on continue la recherche dans la seconde partie du tableau.
- On répète cela jusqu'à avoir trouvé la valeur recherchée, ou bien avoir réduit l'intervalle de recherche à un intervalle vide, ce qui signifie que la valeur recherchée n'est pas présente.

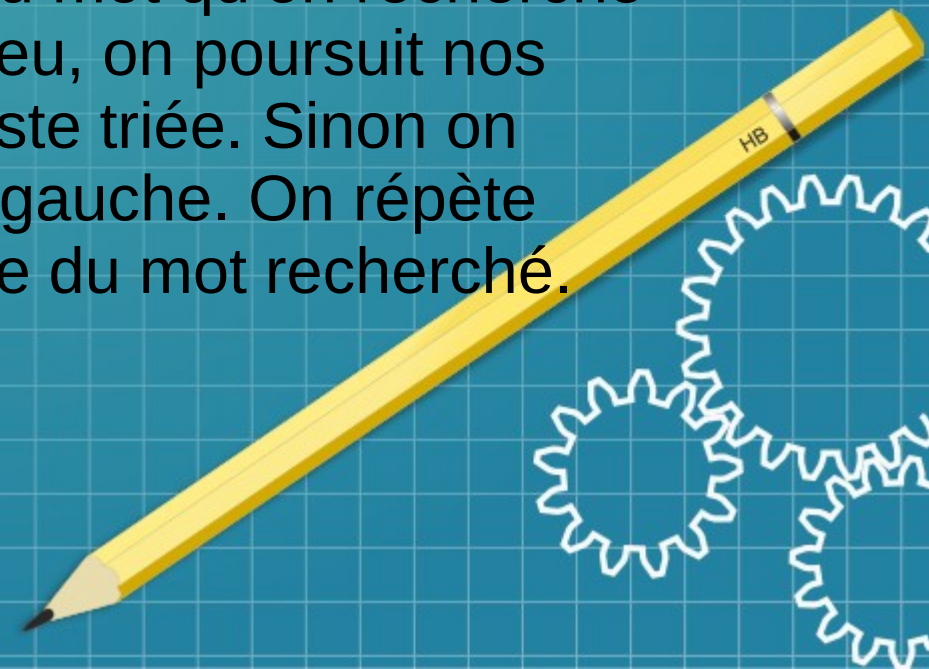


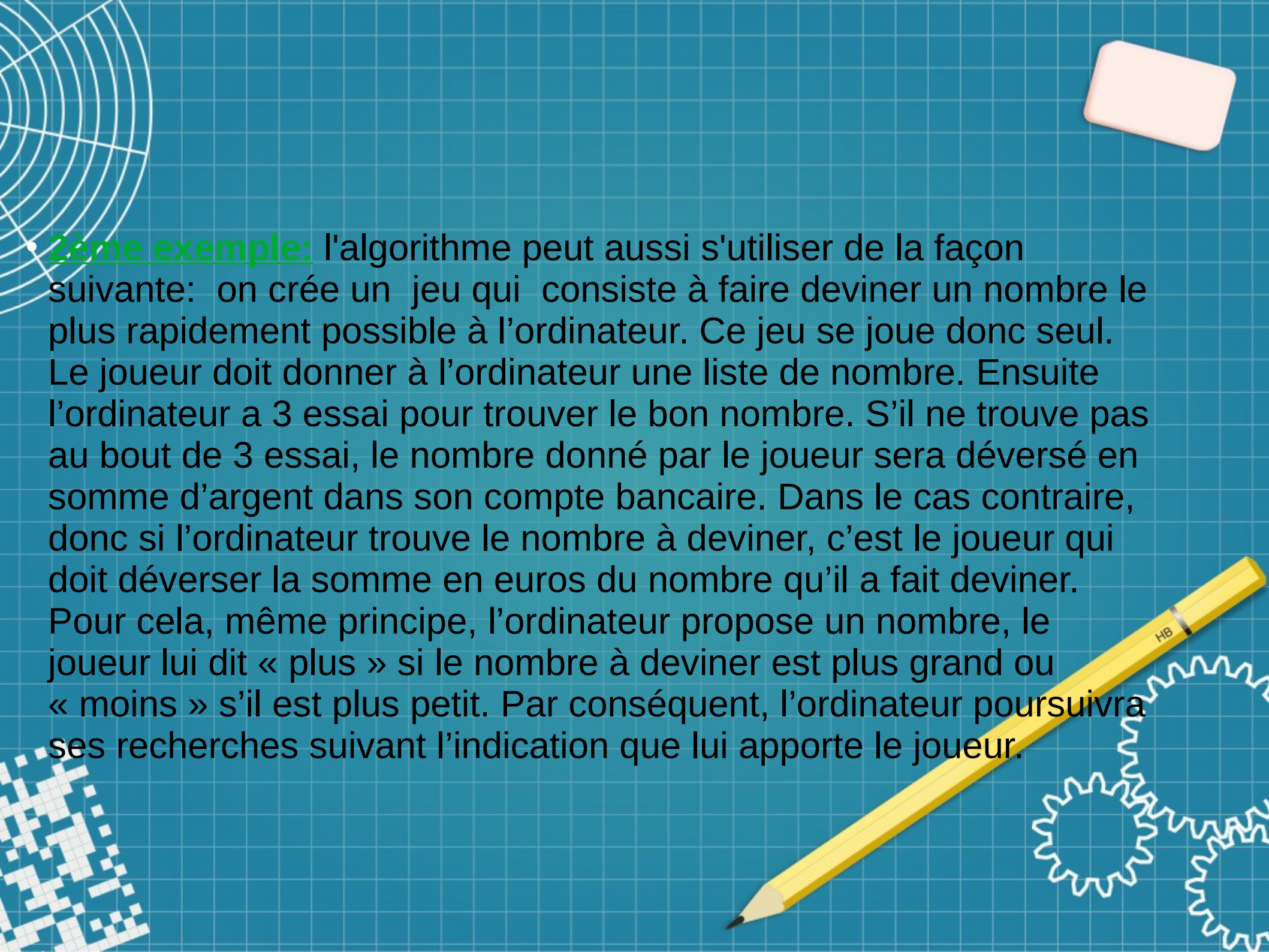
Visualisation d'une recherche dichotomique, où 4 est la valeur recherchée.

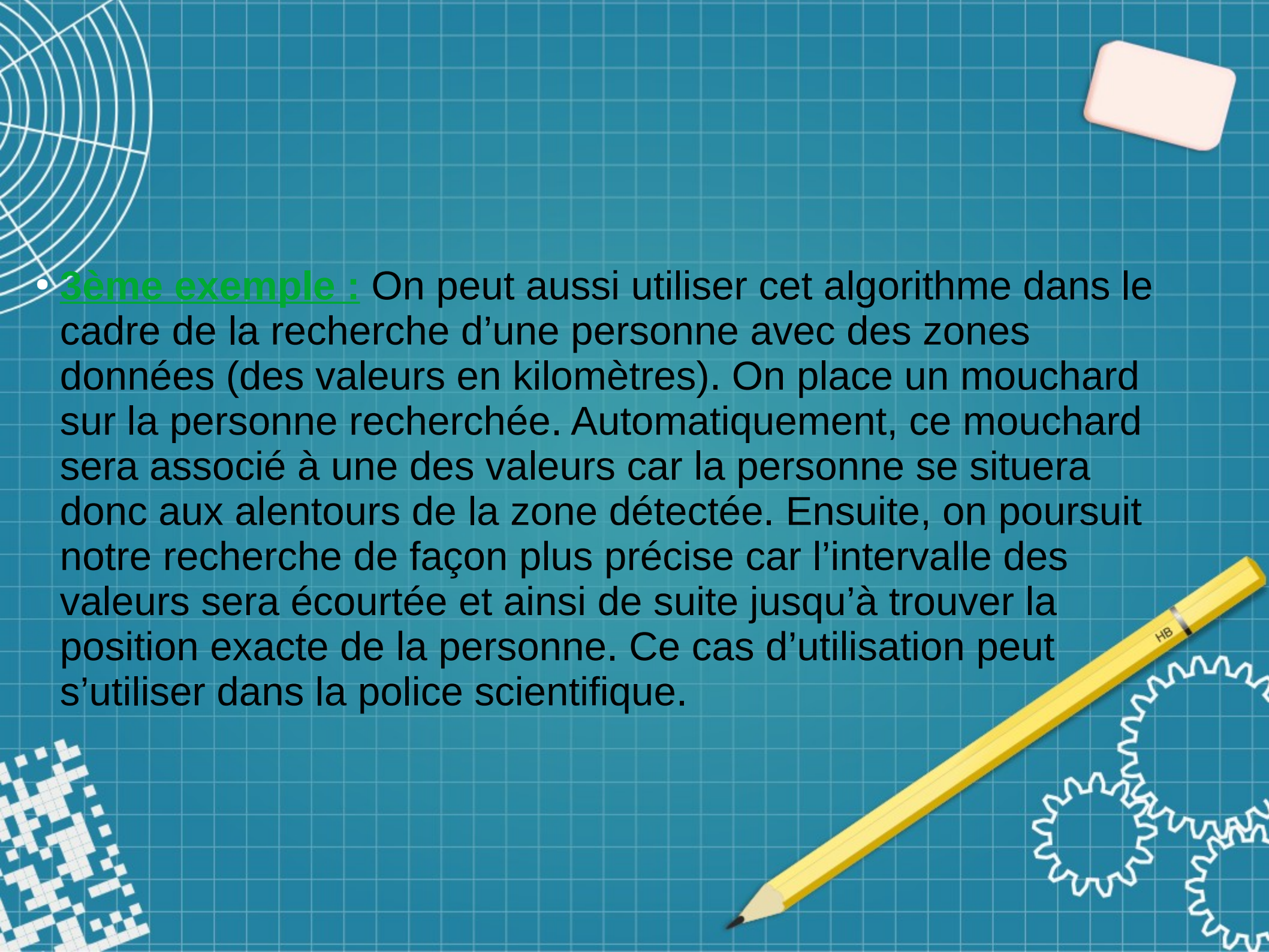
IV/Où utiliser l'algo ?



- 1^{er} exemple: on peut utiliser cet algorithme pour retrouver un mot dans un dictionnaire numérique. On convertit d'abord les 26 lettres de l'alphabet en nombre (le nombre attribué à chacune des lettres sera sa position dans l'alphabet, pour cela on va utiliser un dico (la fonction)). Ensuite, on va se placer au milieu du dictionnaire et si la première lettre du mot qu'on recherche est placée plus loin que la lettre du milieu, on poursuit nos recherches dans la partie droite de la liste triée. Sinon on poursuit nos recherches dans la partie gauche. On répète l'action jusqu'à trouver la première lettre du mot recherché.



- 
- 2ème exemple: l'algorithme peut aussi s'utiliser de la façon suivante: on crée un jeu qui consiste à faire deviner un nombre le plus rapidement possible à l'ordinateur. Ce jeu se joue donc seul. Le joueur doit donner à l'ordinateur une liste de nombre. Ensuite l'ordinateur a 3 essais pour trouver le bon nombre. S'il ne trouve pas au bout de 3 essais, le nombre donné par le joueur sera déversé en somme d'argent dans son compte bancaire. Dans le cas contraire, donc si l'ordinateur trouve le nombre à deviner, c'est le joueur qui doit déverser la somme en euros du nombre qu'il a fait deviner. Pour cela, même principe, l'ordinateur propose un nombre, le joueur lui dit « plus » si le nombre à deviner est plus grand ou « moins » s'il est plus petit. Par conséquent, l'ordinateur poursuivra ses recherches suivant l'indication que lui apporte le joueur.

- 
- 3ème exemple : On peut aussi utiliser cet algorithme dans le cadre de la recherche d'une personne avec des zones données (des valeurs en kilomètres). On place un mouchard sur la personne recherchée. Automatiquement, ce mouchard sera associé à une des valeurs car la personne se situera donc aux alentours de la zone détectée. Ensuite, on poursuit notre recherche de façon plus précise car l'intervalle des valeurs sera écourtée et ainsi de suite jusqu'à trouver la position exacte de la personne. Ce cas d'utilisation peut s'utiliser dans la police scientifique.

V/ Exemple d'algo d'une recherche dichotomique simple

```
def recherche_dichotomique(tab, val):  
    gauche = 0  
    droite = len(tab) - 1  
    while gauche <= droite:  
        milieu = (gauche + droite) // 2  
        if tab[milieu] == val:  
            # on a trouvé val dans le tableau,  
            # à la position milieu  
            return milieu  
        elif tab[milieu] > val:  
            # on cherche entre gauche et milieu - 1  
            droite = milieu - 1  
        else: # on a tab[milieu] < val  
            # on cherche entre milieu + 1 et droite  
            gauche = milieu + 1  
    # on est sorti de la boucle sans trouver val  
    return -1
```



VI) Variant de boucle

- La recherche dichotomique contient une boucle while, on ne connaît donc pas le nombre de répétition que cette boucle va faire (étant donné que c'est une boucle non bornée). Pour être sûr de toujours obtenir un résultat, il faut s'assurer que l'on ne reste pas bloqué infiniment dans la boucle.
- Pour cela, nous allons utiliser un variant de boucle. Un variant de boucle est un nombre entier qui:
 - doit être positif ou nul pour rester dans la boucle ;
 - doit décroître strictement à chaque répétition.
- Lorsqu'on trouve ce variant, on va obligatoirement sortir de la boucle au bout d'un nombre fini de répétitions, puisque un entier positif ne peut décroître infiniment.

Preuve

- Tout d'abord pour montrer que le variant décroît lors de l'exécution de la boucle, on commence par définir que : $\text{milieu} = (\text{gauche} + \text{droite}) // 2$.
- En particulier, on a alors $\text{gauche} \leq \text{milieu} \leq \text{droite}$.
- Ensuite, on a trois cas qui sont possibles.
- Si $\text{tab}[\text{milieu}] == \text{val}$, on sort directement de la boucle à l'aide d'un return. La terminaison est assurée.
- Si $\text{tab}[\text{milieu}] > \text{val}$, on modifie la valeur de droite. En appelant droite2 cette nouvelle valeur, on a :
$$\text{droite2} - \text{gauche} < \text{milieu} - \text{gauche} \leq \text{droite} - \text{gauche}$$

car $\text{droite2} = \text{milieu} - 1 < \text{milieu}$.
Ainsi, le variant décroît.
- Sinon, on modifie gauche et on a de même :
$$\text{droite} - \text{gauche2} < \text{droite} - \text{milieu} \leq \text{droite} - \text{gauche}$$

De même, le variant décroît.
- (se référer à l'exemple proposé 2 pages plus tôt)