Nom :	Prénom :	Classe :

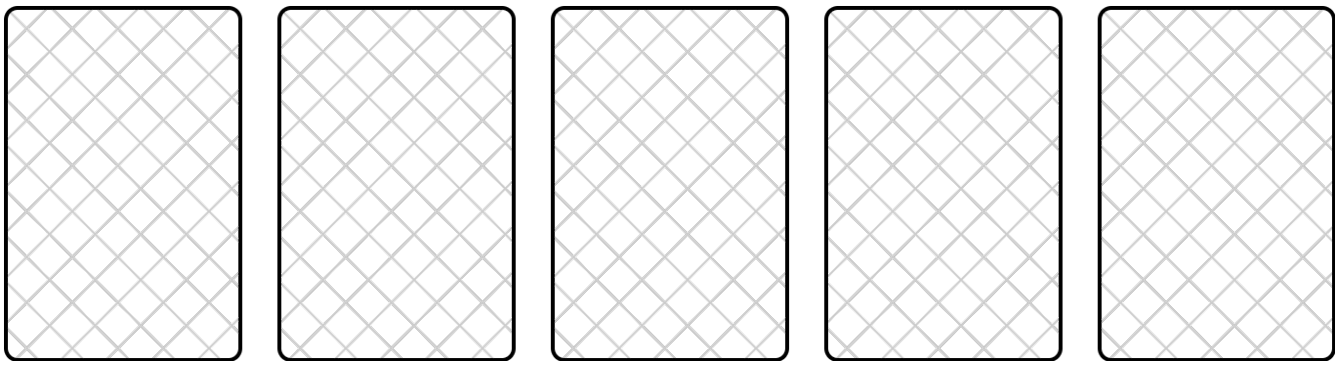
NSI 1re — Algorithme de tri par insertion

Objectifs :
- Savoir **reconnaître** un algorithme de tri par insertion (d'un tableau indexé).
- Savoir en expliquer **le fonctionnement**, les principes.
- Savoir **écrire un algorithme de tri**.

Introduction

Souvent, dans la vie, nous réorganisons des *choses*, nous trions des *affaires*, sans forcément faire attention à la méthode (l'algorithme) que nous employons pour trier.

Prenons par exemple cinq cartes au hasard d'un paquet de 32 cartes, dont les valeurs vont de 1 à 32.



Ces cartes sont disposées face cachée, et nous aimerions les trier dans un ordre précis. Par exemple, dans un ordre croissant. Nous savons que chaque carte peut avoir une valeur comprise entre 1 et 32.

➤ **Exercice 1** — Deviner

Comment procéderiez-vous pour trier les cinq cartes de l'exemple ci-dessus, *comme un robot* ?

Tri par insertion

Le tri par insertion parcourt un tableau de la gauche vers la droite, en gardant sur la gauche une *zone déjà triée* :

zone déjà triée	zone pas encore triée

À chaque étape, le tri par insertion va insérer l'élément le plus à gauche de la *zone pas encore triée*, dans la la *zone déjà triée*.

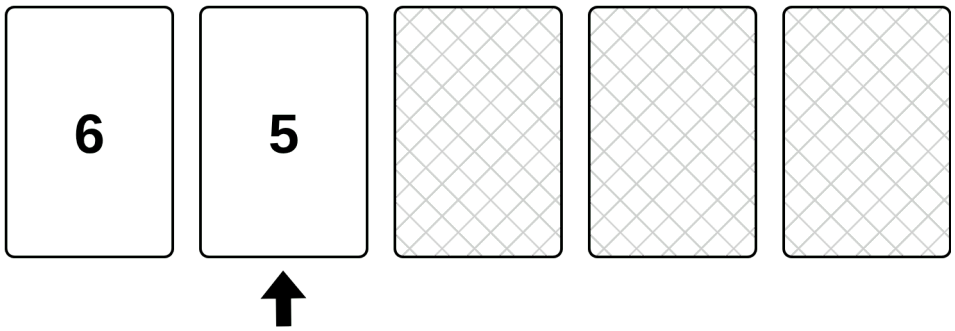
Pour cela, on va décaler d'une case vers la droite tous les éléments déjà triés qui sont plus grands que la valeur à insérer, puis déposer cette dernière dans la case ainsi libérée.

Déroulé visuel

Voici un exemple de *déroulé visuel*, où l'on effectue un tri par insertion (dans l'ordre croissant) de cinq cartes :

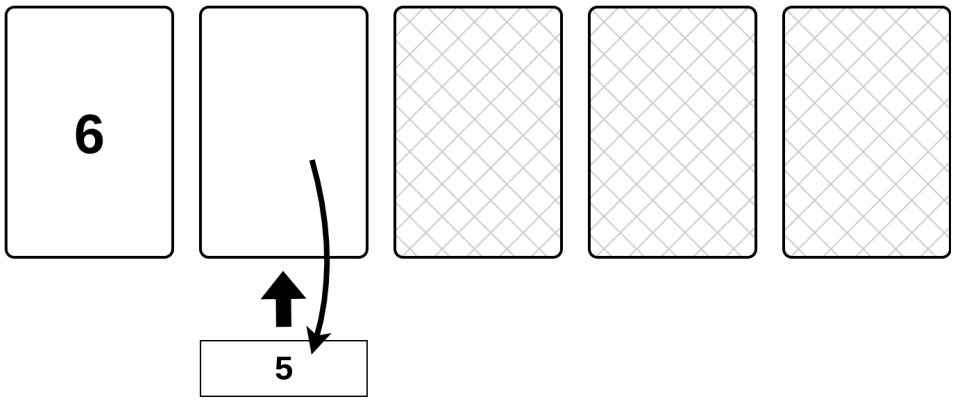
étape 1

on commence par le deuxième élément



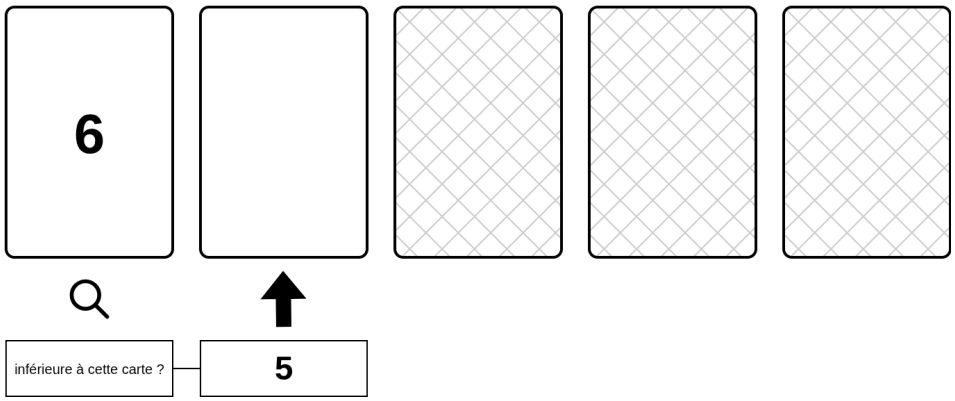
étape 2

on met de côté sa valeur (valeur clé), tout en rendant son emplacement disponible



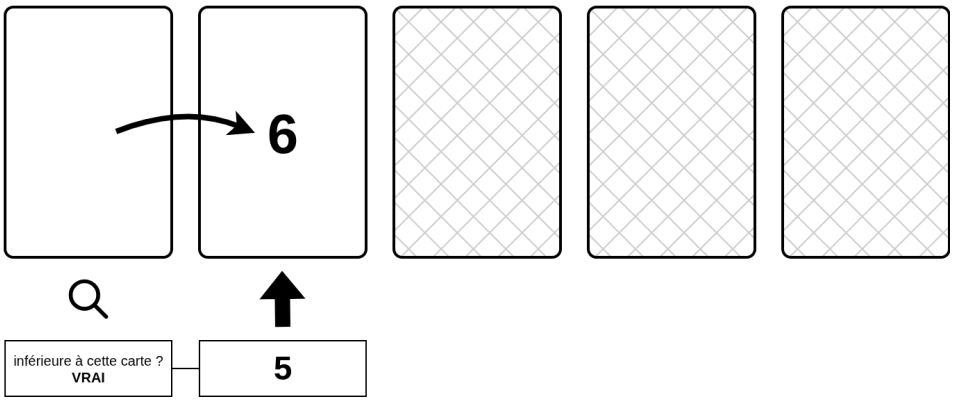
étape 3

on compare la valeur clé avec celle de la carte précédente



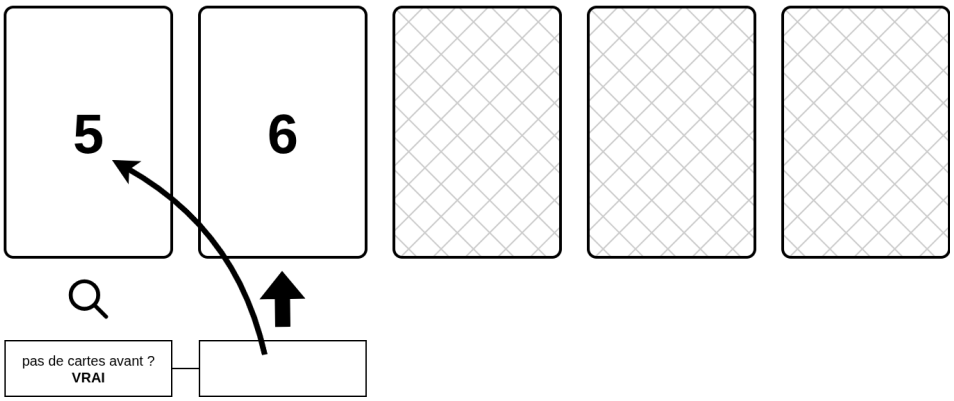
étape 4

la valeur clé est inférieure à cette carte observée, donc on la déplace d'un cran vers la droite



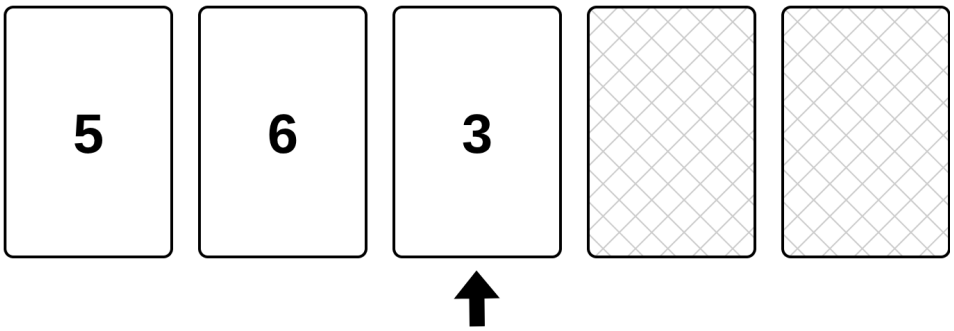
étape 5

on est arrivé au début du paquet, la comparaison avec la valeur clé s'arrête ici. On stocke la valeur clé à l'emplacement disponible



étape 6

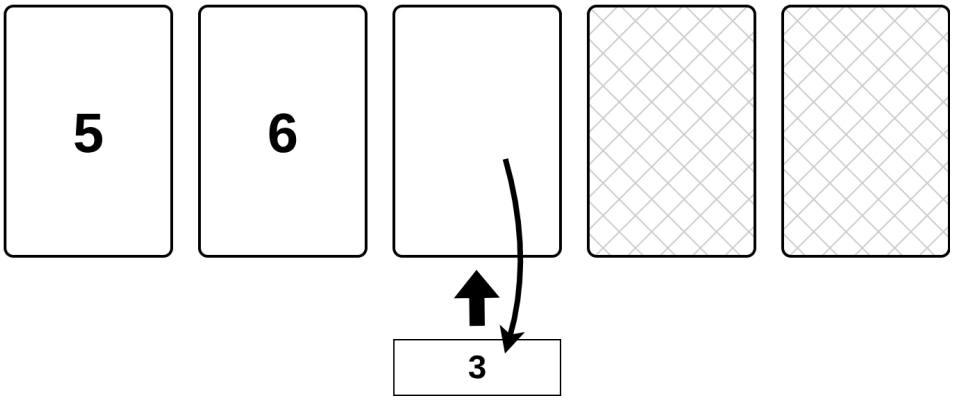
on s'occupe de l'élément suivant



étape 7

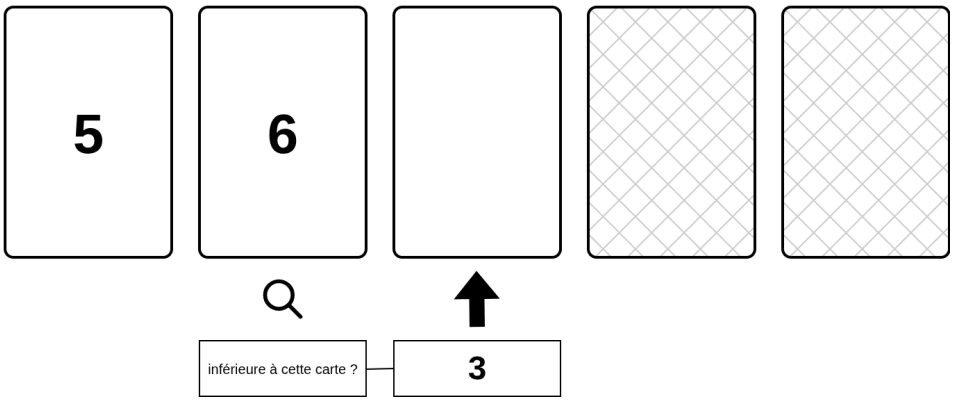
on met de côté sa valeur, c'est ce qu'on appelle la "valeur clé"

en faisant cela on libère une place



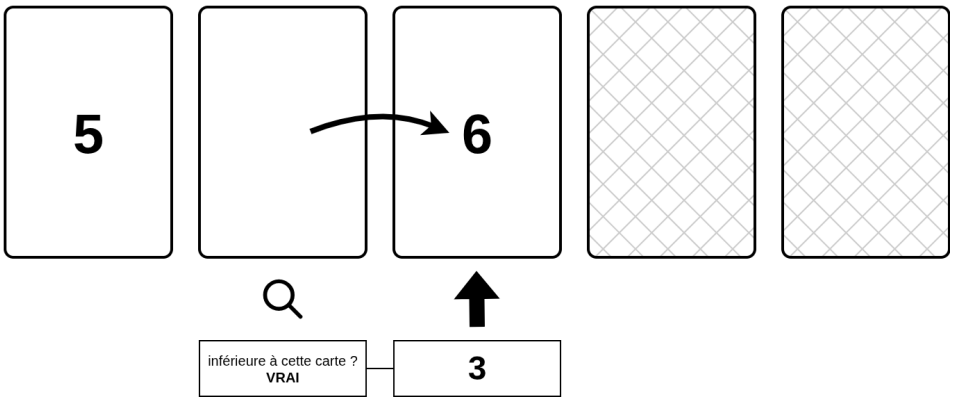
étape 8

on compare la valeur clé à la carte précédente



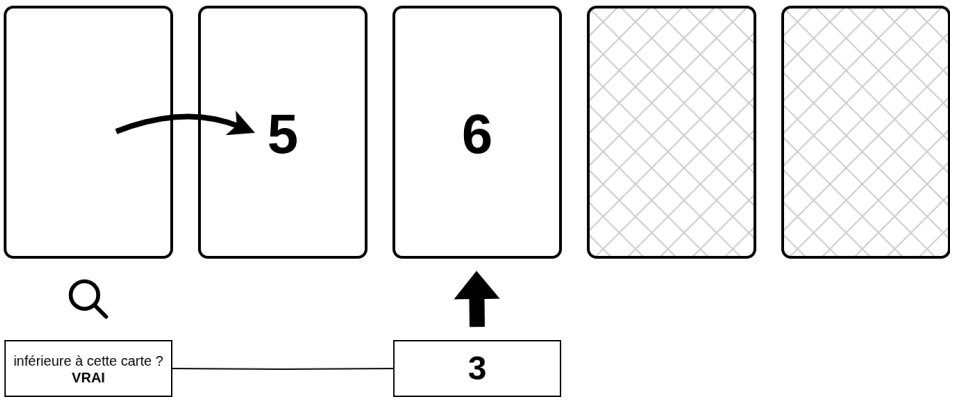
étape 9

notre valeur clé est inférieure, donc on décale cette carte observée d'un cran à droite



étape 10

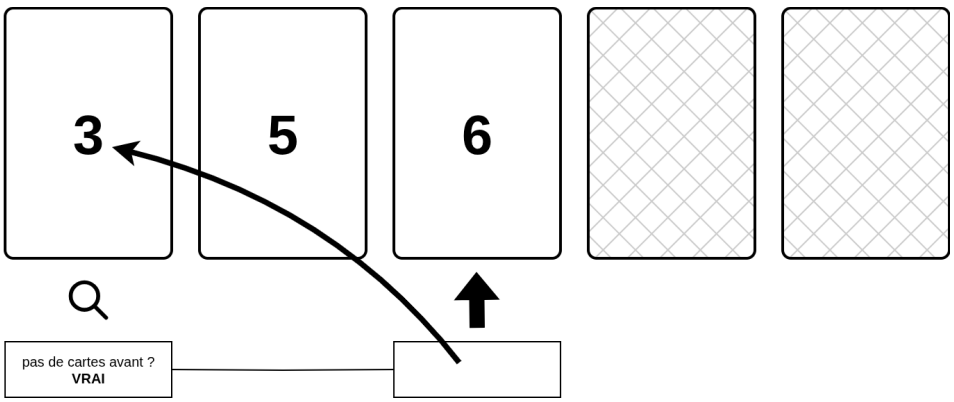
notre valeur clé est également inférieure à cette carte



étape 11

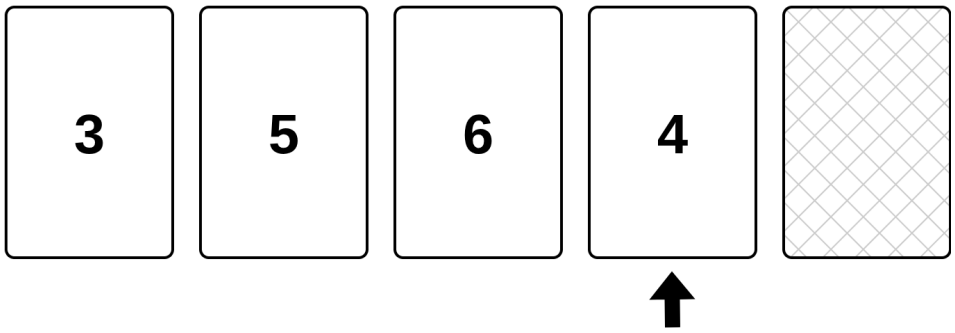
on est arrivé au début du paquet, la comparaison avec la valeur clé s'arrête ici.

On stocke la valeur clé à l'emplacement disponible



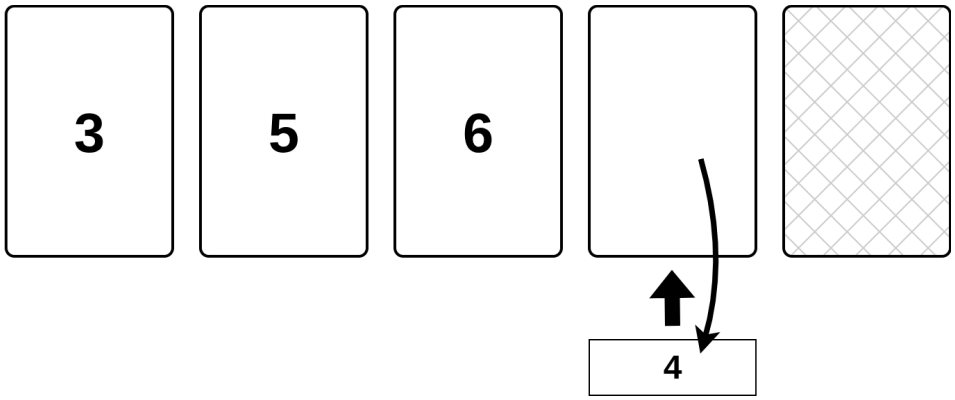
étape 12

on continue avec la prochaine carte



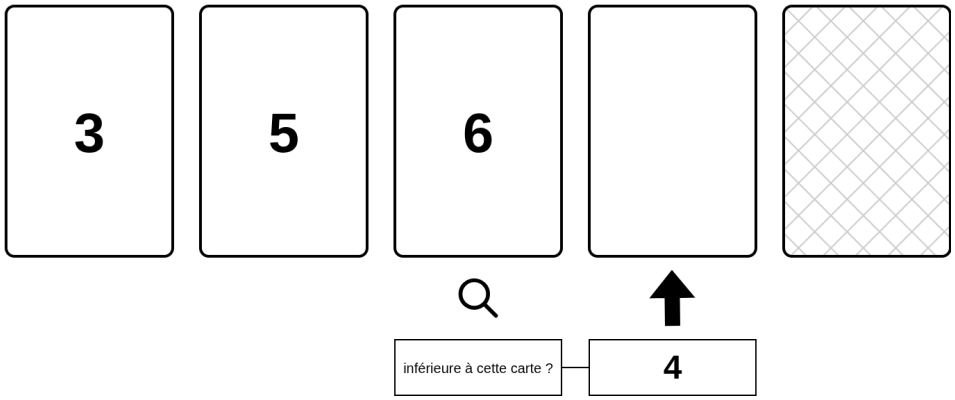
étape 13

on stocke sa valeur comme "valeur clé", et on libère une place



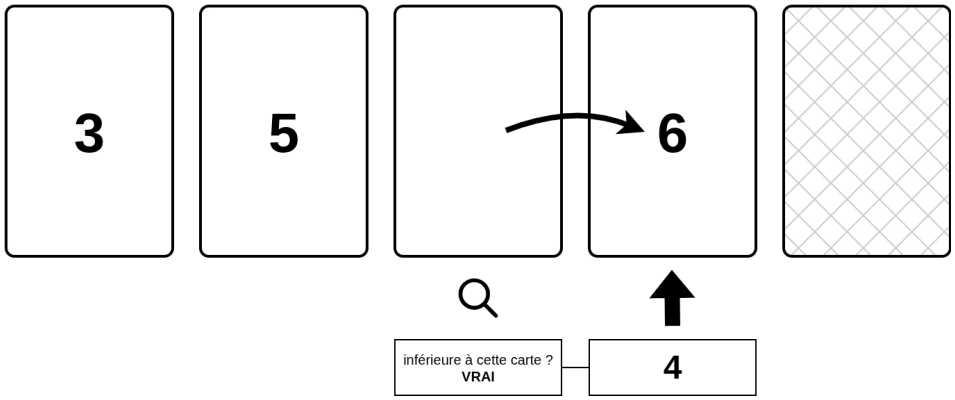
étape 14

on compare la valeur clé aux cartes précédentes

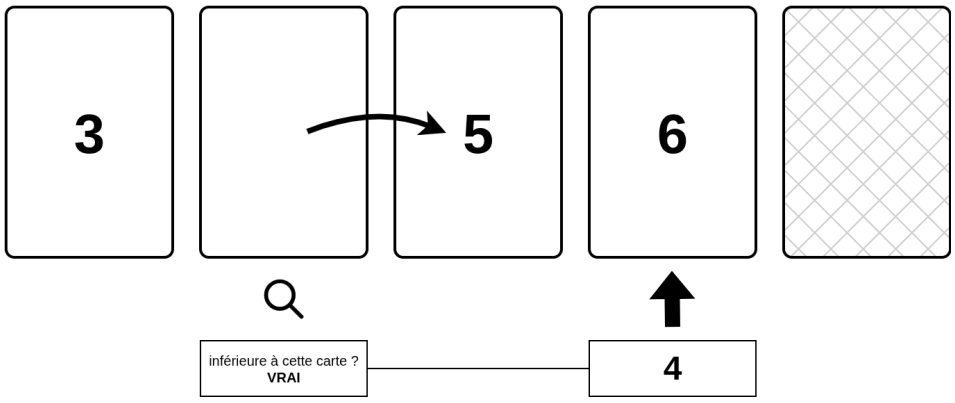


étape 15

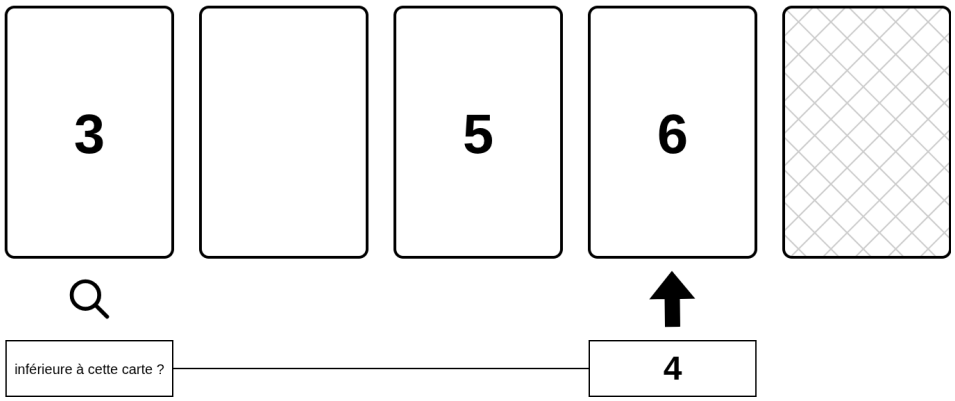
etc.



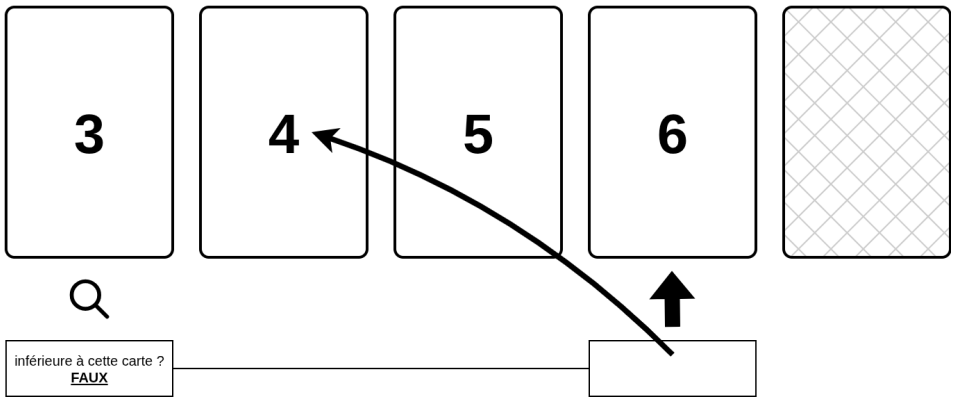
étape 16



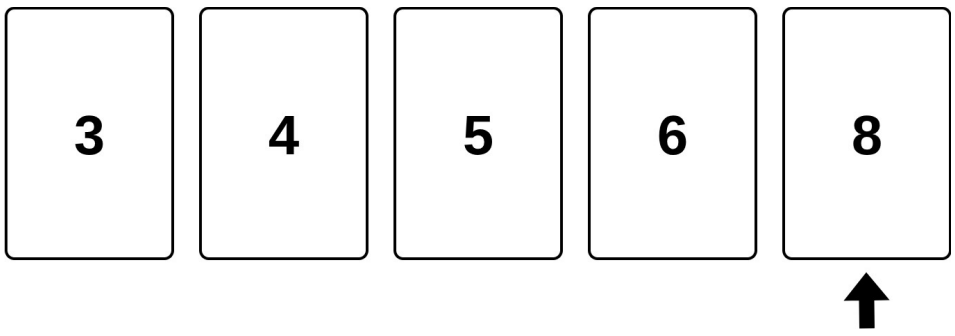
étape 17



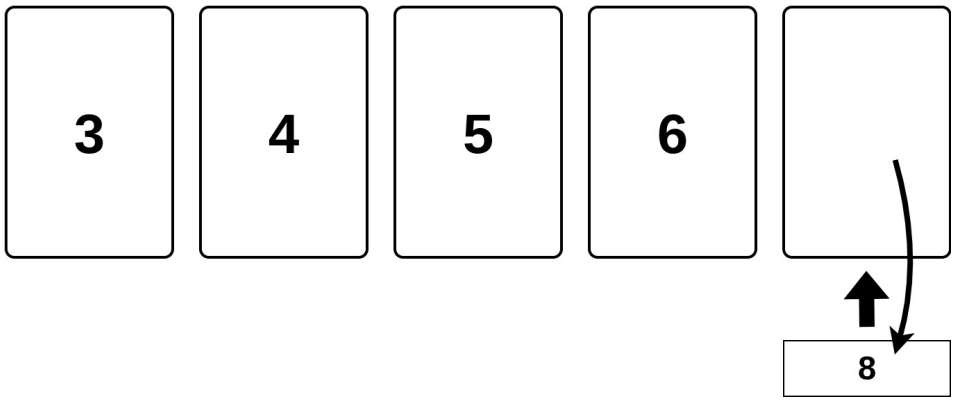
étape 18



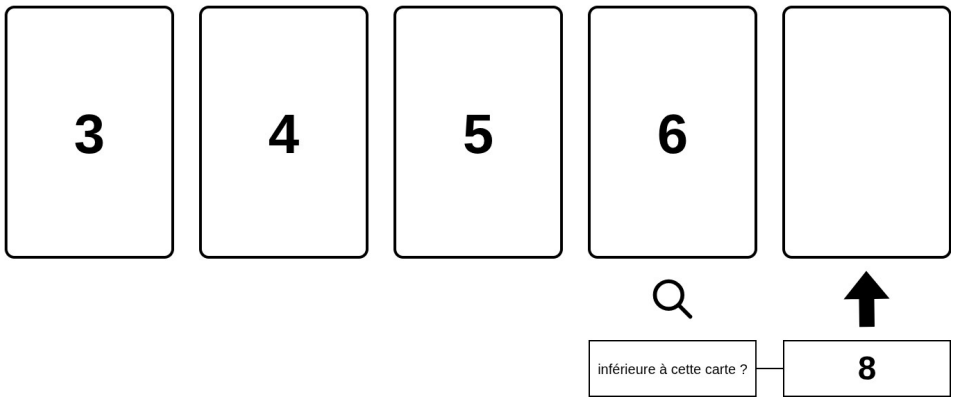
étape 19



étape 20



étape 21

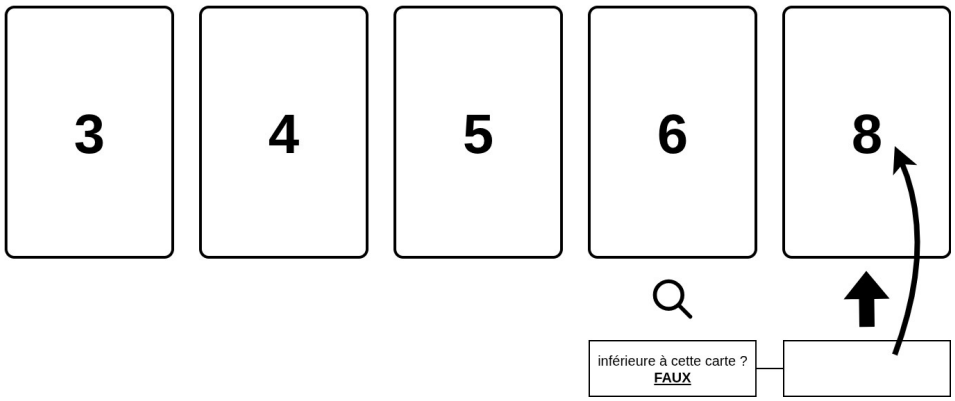


étape 22

cette valeur clé n'est pas inférieure à la carte observée.

La comparaison avec la valeur clé s'arrête ici.

On stocke la valeur clé à l'emplacement disponible



Déroulé détaillé

Il est également possible de représenter les étapes d'un tri par insertion (ordre croissant) sous la forme d'un *déroulé détaillé* :

	Carte	Carte	Carte	Carte	Carte
	6	5	3	4	8
Valeur actuelle	6	5	3	4	8
Place libérée	6		3	4	8
Décalage		6	3	4	8
Dépôt	5	6	3	4	8
Valeur actuelle	5	6	3	4	8
Place libérée	5	6		4	8
Décalage	5		6	4	8
Décalage		5	6	4	8
Dépôt	3	5	6	4	8
Valeur actuelle	3	5	6	4	8
Place libérée	3	5	6		8
Décalage	3	5		6	8
Décalage	3		5	6	8
Dépôt	3	4	5	6	8
Valeur actuelle	3	4	5	6	8
Place libérée	3	4	5	6	
Dépôt	3	4	5	6	8
Fin	3	4	5	6	8

➤ **Exercice 2** — Enquêter

À vous de jouer, complétez les étapes d'un tri par insertion (ordre croissant) des éléments 21, 10 et 4, sous la forme d'un *déroulé détaillé* :

	Carte	Carte	Carte
	21	10	4
Valeur actuelle	…	…	…
Place libérée	…	…	…
Décalage	…	…	…
Dépôt	…	…	…
Valeur actuelle	…	…	…
Place libérée	…	…	…
Décalage	…	…	…
Décalage	…	…	…
Dépôt	…	…	…
Fin	…	…	…

Déroulé compact

Et enfin, voici une représentation des étapes d'un tri par insertion (ordre croissant) sous la forme d'un *déroulé compact*. Ici, en prenant par exemple un tableau contenant les éléments 10, 8, 3 :

10	8	3
10		3
	10	3
8	10	3
8	10	3
8	10	
8		10
	8	10
3	8	10

➤ **Exercice 3** — Enquêter

À vous de jouer, complétez les étapes d'un tri par insertion (ordre croissant) sous la forme d'un *déroulé compact* du tableau contenant les éléments 23, 10, 15, 12 :

23	10	15	12
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…

Implémentation en pseudo-code

Comme tout algorithme, le tri par insertion peut être implémenté dans n'importe quel langage de programmation. Une première approche pourrait être d'analyser une implémentation possible en *pseudo-code*[1] :

Tri dans l'ordre croissant d'un tableau indexé :

```
Pour i de 1 à Taille(tableau)
    cle = tableau[i]
    j = i
    Tant que tableau[j - 1] > cle et j > 0
        tableau[j] = tableau[j - 1]
        j = j - 1
    tableau[j] = cle
```

Implémentation en Python

➤ Exercice 4 — Enquêter

Convertissez le pseudo-code du tri par insertion dans l'ordre croissant en langage Python :

```

```

1 Un pseudo-code, c’est une manière simple d’écrire les étapes d’un programme sans utiliser un vrai langage de programmation.

Pour aller plus loin

N'hésitez pas à consulter cette petite vidéo expliquant le tri par insertion :

https://fmr.tf/s/le_tri_par_insertion.mp4

🌶️ Supplément "chili pepper" — facultatif

➤ **Exercice 5** — Modifier

Convertissez le pseudo-code du tri par insertion dans l'ordre décroissant en langage Python :