Introduction à l’informatique CM5

Antonio E. Porreca
aeporreca.org/introinfo

⚠️ Rappel ⚠️

Vendredi 23 octobre à 13h
on a le partiel !

Recherche dans un annuaire ou un dictionnaire ?

Recherche dichotomique dans un tableau d’entiers trié

```
fonction chercher(x, T)
    n ≔ longueur(T)
    i ≔ 0
    j ≔ n – 1
    tant que i ≤ j faire
        m ≔ (i + j) ÷ 2
        si T[m] = x alors
            retourner m
        sinon si x < T[m] alors
            j ≔ m – 1
        sinon
            i ≔ m + 1
        fin si
    fin tant que
    retourner –1
fin fonction
```

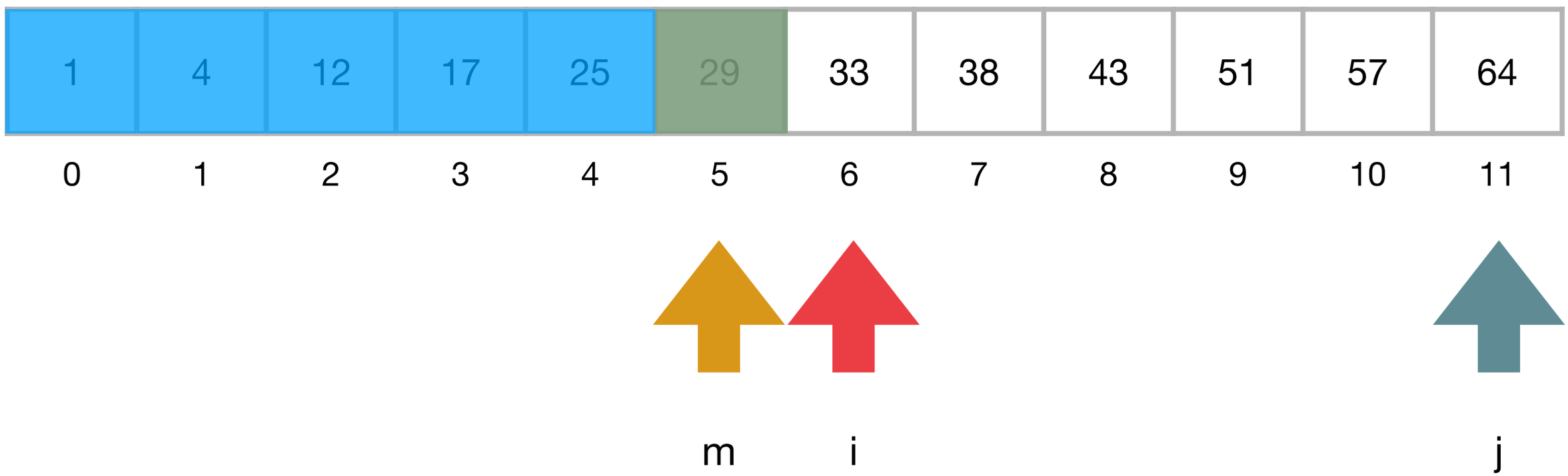
Recherche dichotomique

Recherche de 33

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11
i					m						j

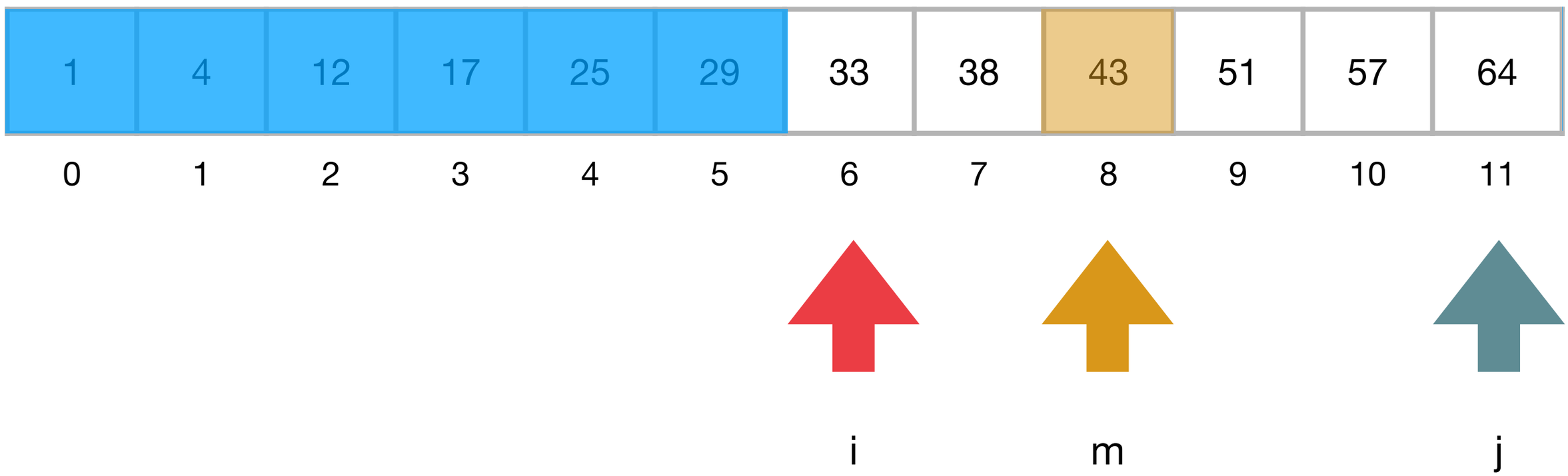
Recherche dichotomique

Recherche de 33



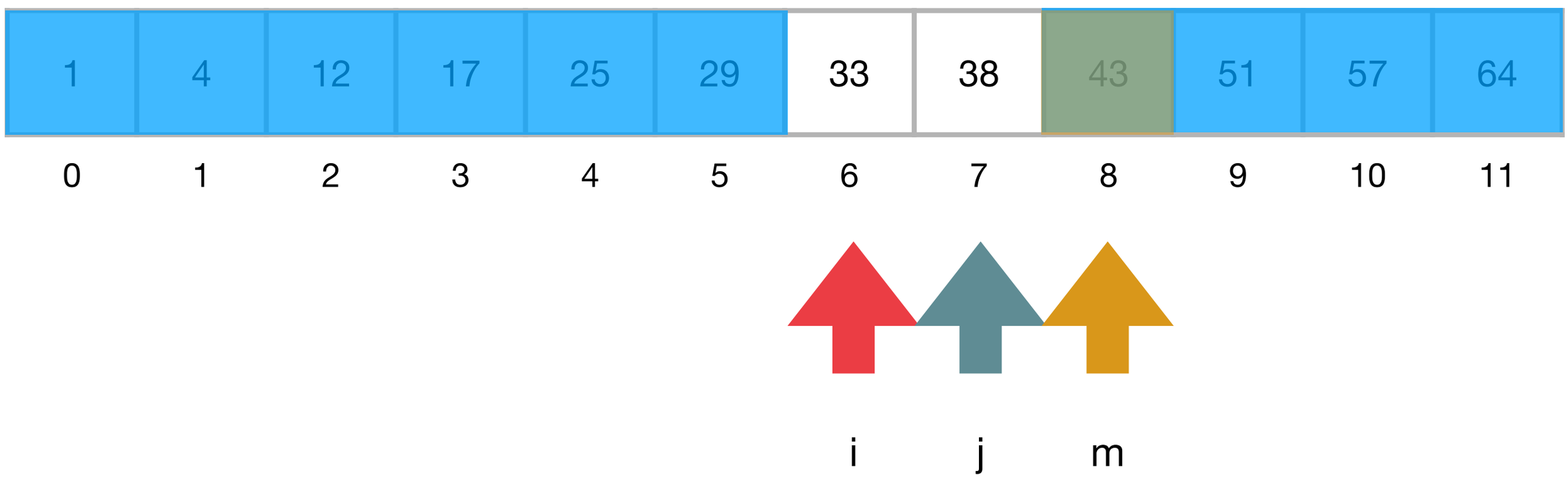
Recherche dichotomique

Recherche de 33



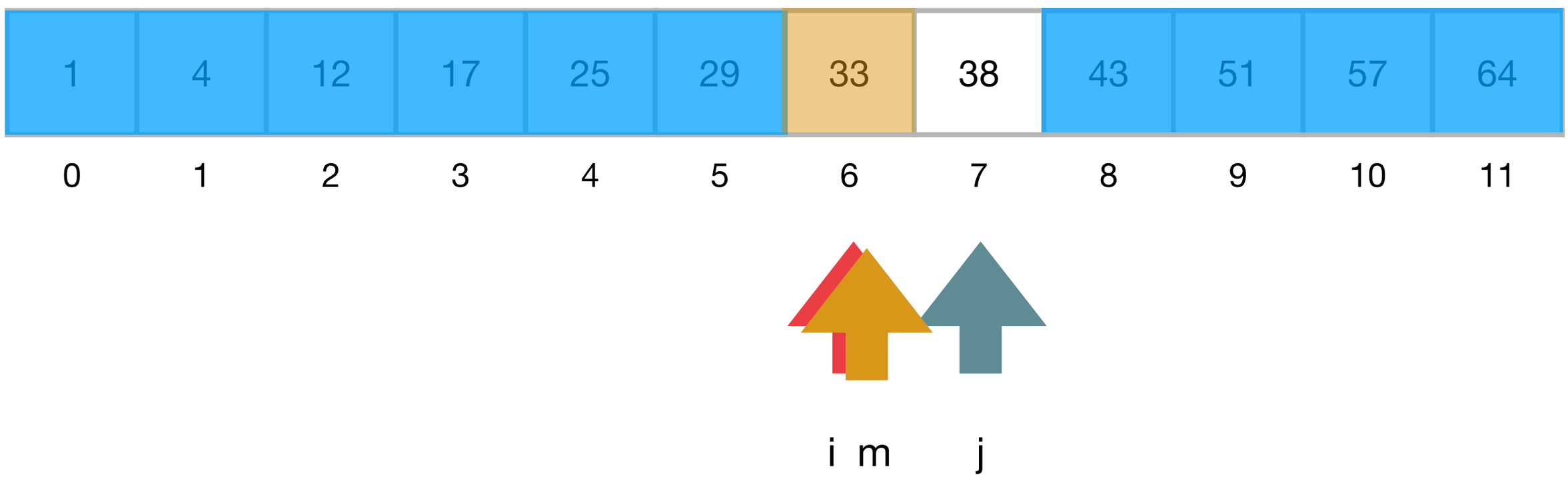
Recherche dichotomique

Recherche de 33



Recherche dichotomique

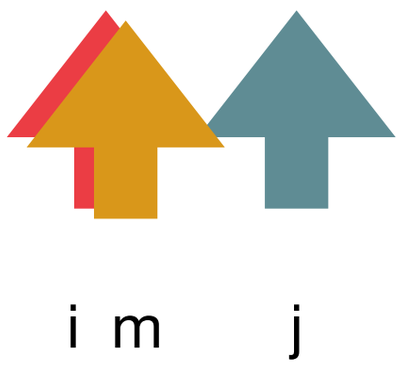
Recherche de 33



Recherche dichotomique

Recherche de 33

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11



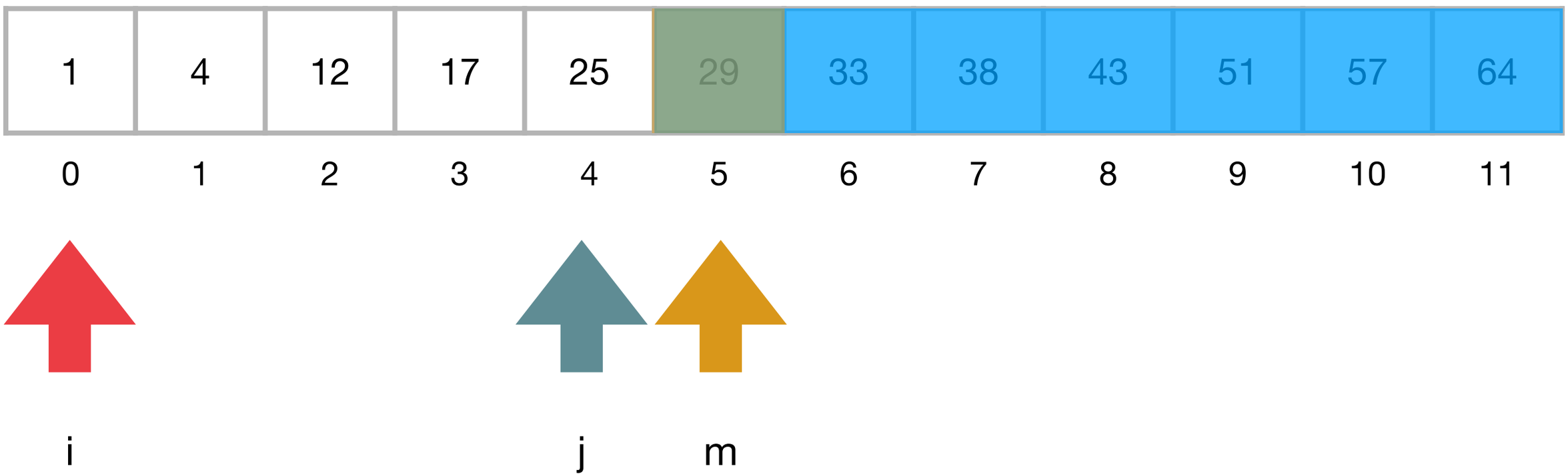
Recherche dichotomique

Recherche de 16

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11
i					m						j

Recherche dichotomique

Recherche de 16



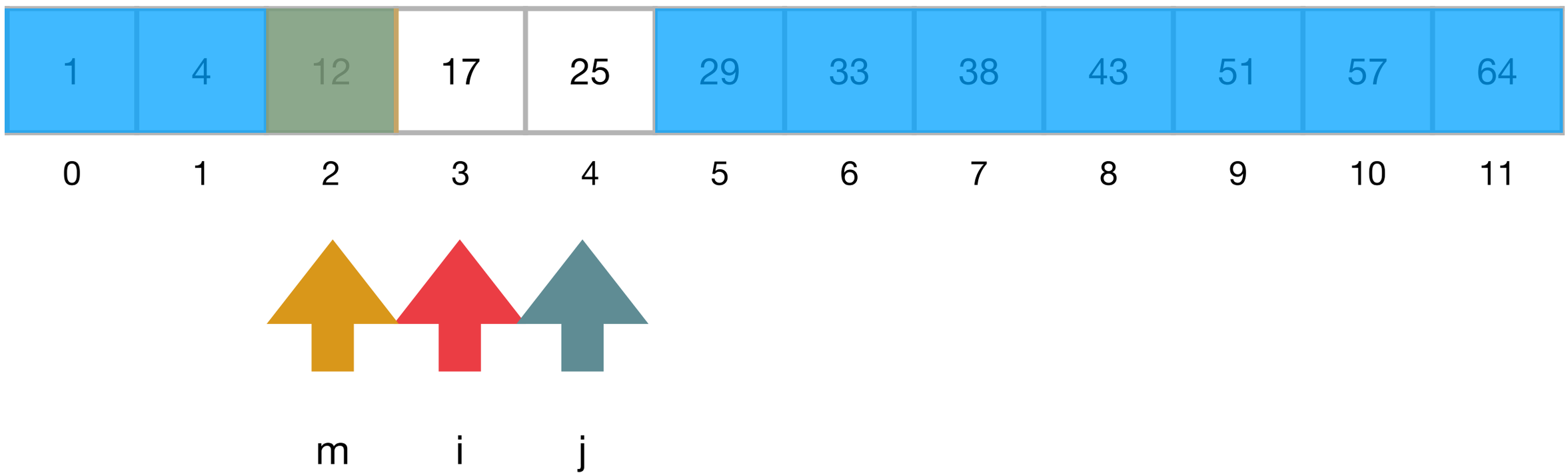
Recherche dichotomique

Recherche de 16

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11
i		m		j							

Recherche dichotomique

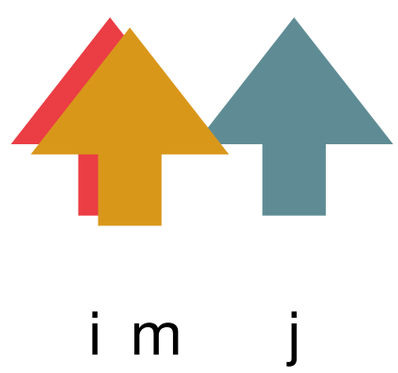
Recherche de 16



Recherche dichotomique

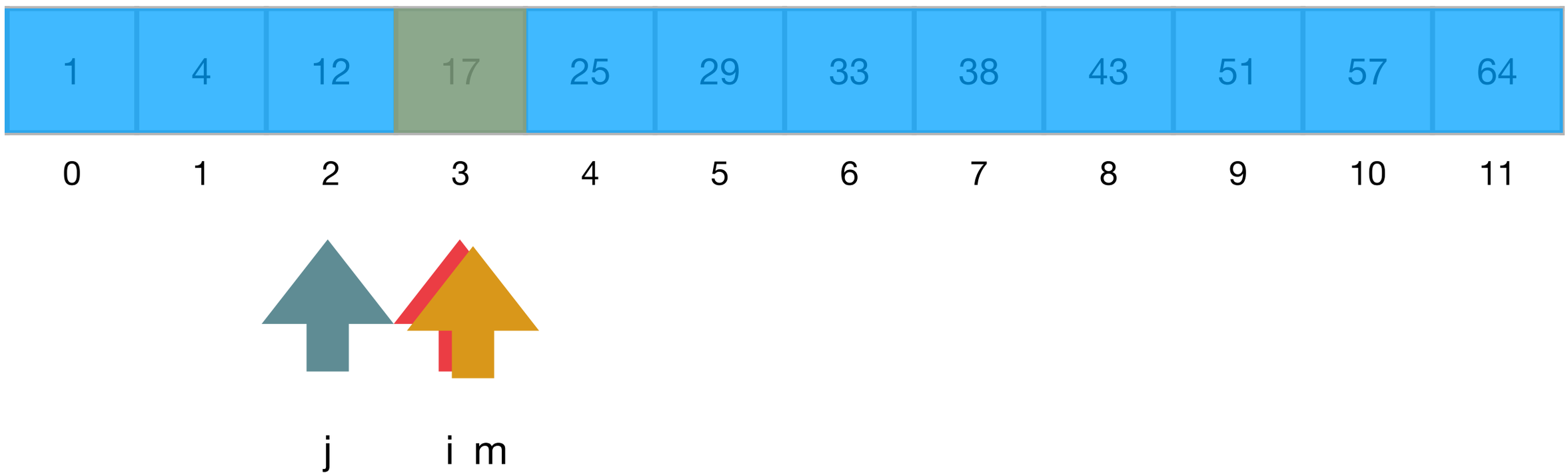
Recherche de 16

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11



Recherche dichotomique

Recherche de 16



Recherche dichotomique

Recherche de 16

1	4	12	17	25	29	33	38	43	51	57	64
0	1	2	3	4	5	6	7	8	9	10	11

j i m

Recherche dichotomique dans un tableau d'entiers trié

```
fonction chercher(x, T)
    n ≔ longueur(T)
    i ≔ 0
    j ≔ n – 1
    tant que i ≤ j faire
        m ≔ (i + j) ÷ 2
        si T[m] = x alors
            retourner m
        sinon si x < T[m] alors
            j ≔ m – 1
        sinon
            i ≔ m + 1
        fin si
    fin tant que
    retourner –1
fin fonction
```

Terminaison ?

Correction ?

Efficacité ?

Terminaison

```
fonction chercher(x, T)
    n ≔ longueur(T)
    i ≔ 0
    j ≔ n – 1
    tant que i ≤ j faire
        m ≔ (i + j) ÷ 2
        si T[m] = x alors
            retourner m
        sinon si x < T[m] alors
            j ≔ m – 1
        sinon
            i ≔ m + 1
        fin si
    fin tant que
    retourner –1
fin fonction
```

- Il reste toujours j – i + 1 éléments à examiner
- À chaque itération, on élimine approx. la moitié des éléments qui restent
- Tôt ou tard on trouve x, ou on reste sans éléments, et l'algorithme termine

Correction

```
fonction chercher(x, T)
    n ≔ longueur(T)
    i ≔ 0
    j ≔ n – 1
    tant que i ≤ j faire
        m ≔ (i + j) ÷ 2
        si T[m] = x alors
            retourner m
        sinon si x < T[m] alors
            j ≔ m – 1
        sinon
            i ≔ m + 1
        fin si
    fin tant que
    retourner –1
fin fonction
```

- Invariant de boucle : si x est dans le tableau, il se trouve dans le sous-tableau T[i, …, j]
  - C'est vrai au début de l'algorithme
  - Ça reste vrai à chaque itération de la boucle, parce qu'on vérifie toujours si T[m] = x ou T[m] > x ou T[m] < x
- Si on sort de la boucle avec i > j, alors si x est dans le tableau, il est dans le sous-tableau vide T[i, j], c'est à dire qu'il n'est pas là

Efficacité

```
fonction chercher(x, T)
    n ≔ longueur(T)
    i ≔ 0
    j ≔ n – 1
    tant que i ≤ j faire
        m ≔ (i + j) ÷ 2
        si T[m] = x alors
            retourner m
        sinon si x < T[m] alors
            j ≔ m – 1
        sinon
            i ≔ m + 1
        fin si
    fin tant que
    retourner –1
fin fonction
```

- Dans le pire des cas, x n'est pas là
- Comme on élimine à chaque itération la moitié du tableau, on exécute la boucle $\log_2 n$ fois au maximum
- Ça fait $O(\log_2 n)$ opérations

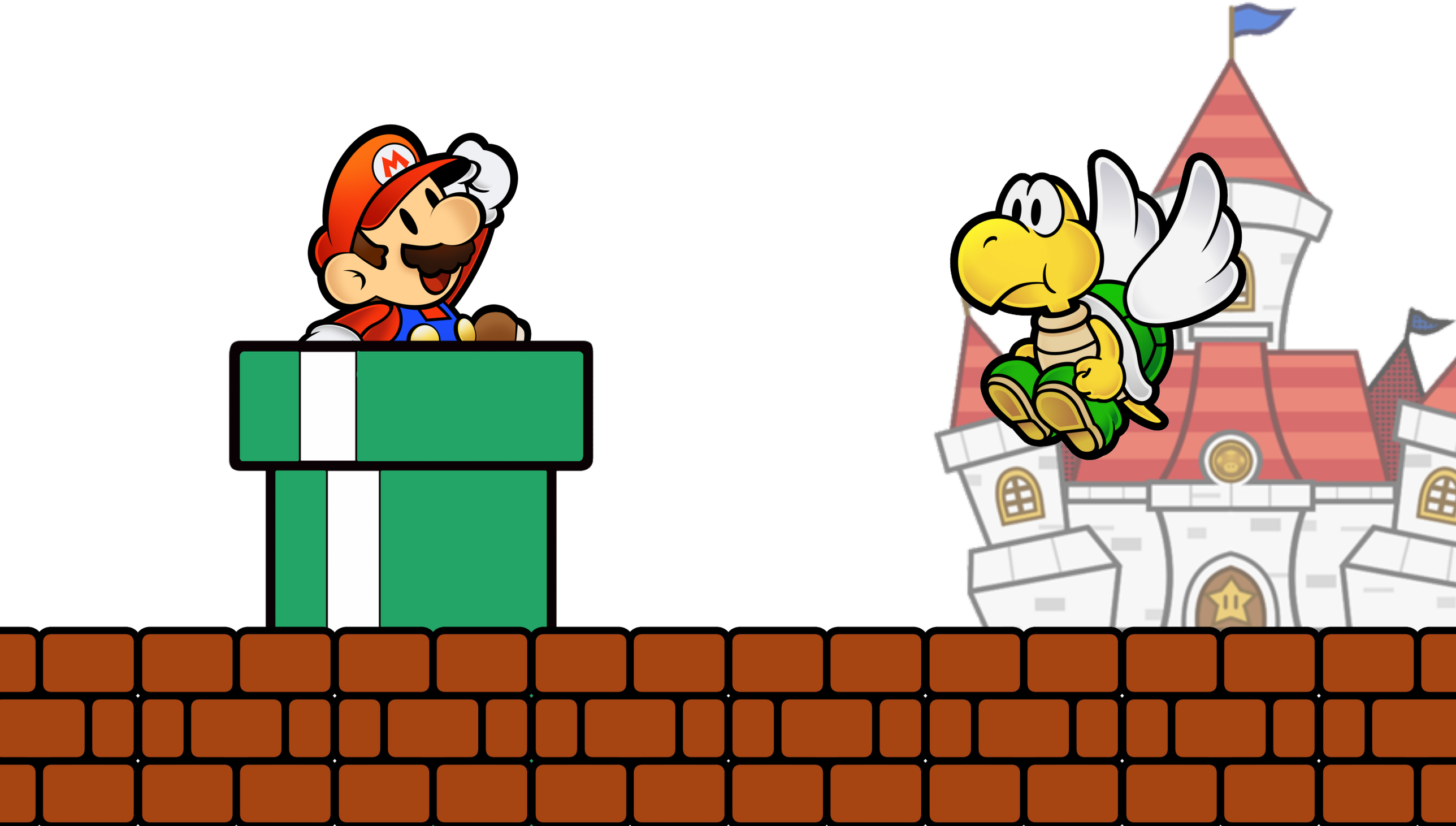
Algorithmes de tri

Algorithmes de tri pour accélérer la recherche dans un tableau

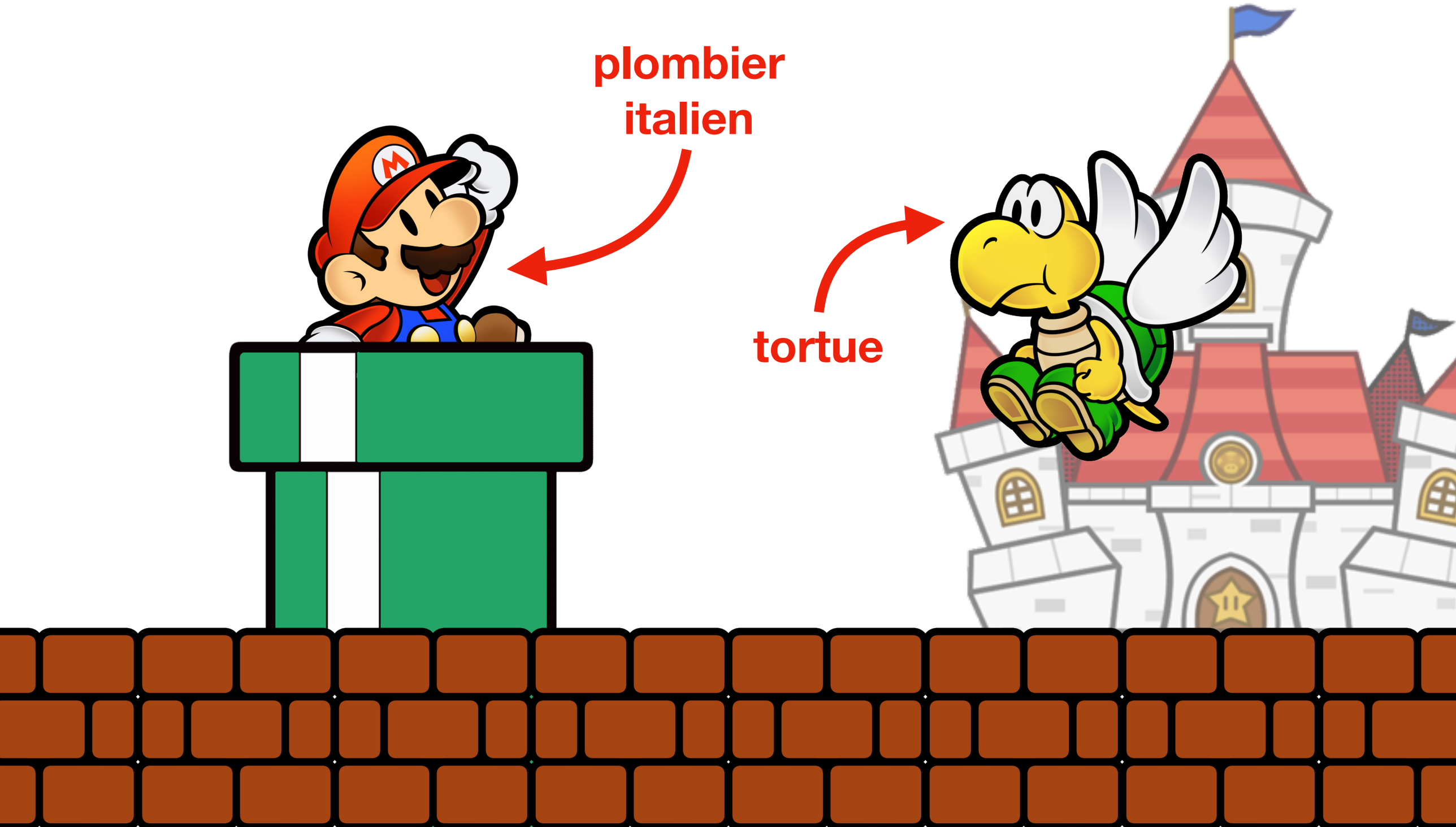
- La recherche dans un tableau non trié prend temps O(n) avec la recherche séquentielle (ou linéaire)
- Par contre, on peut faire une recherche dichotomique dans un tableau trié en temps $O(\log_2 n)$
- Donc ça vaut la peine de trier le tableau si on a beaucoup de recherches à faire

Algorithmes de tri dans le commerce électronique

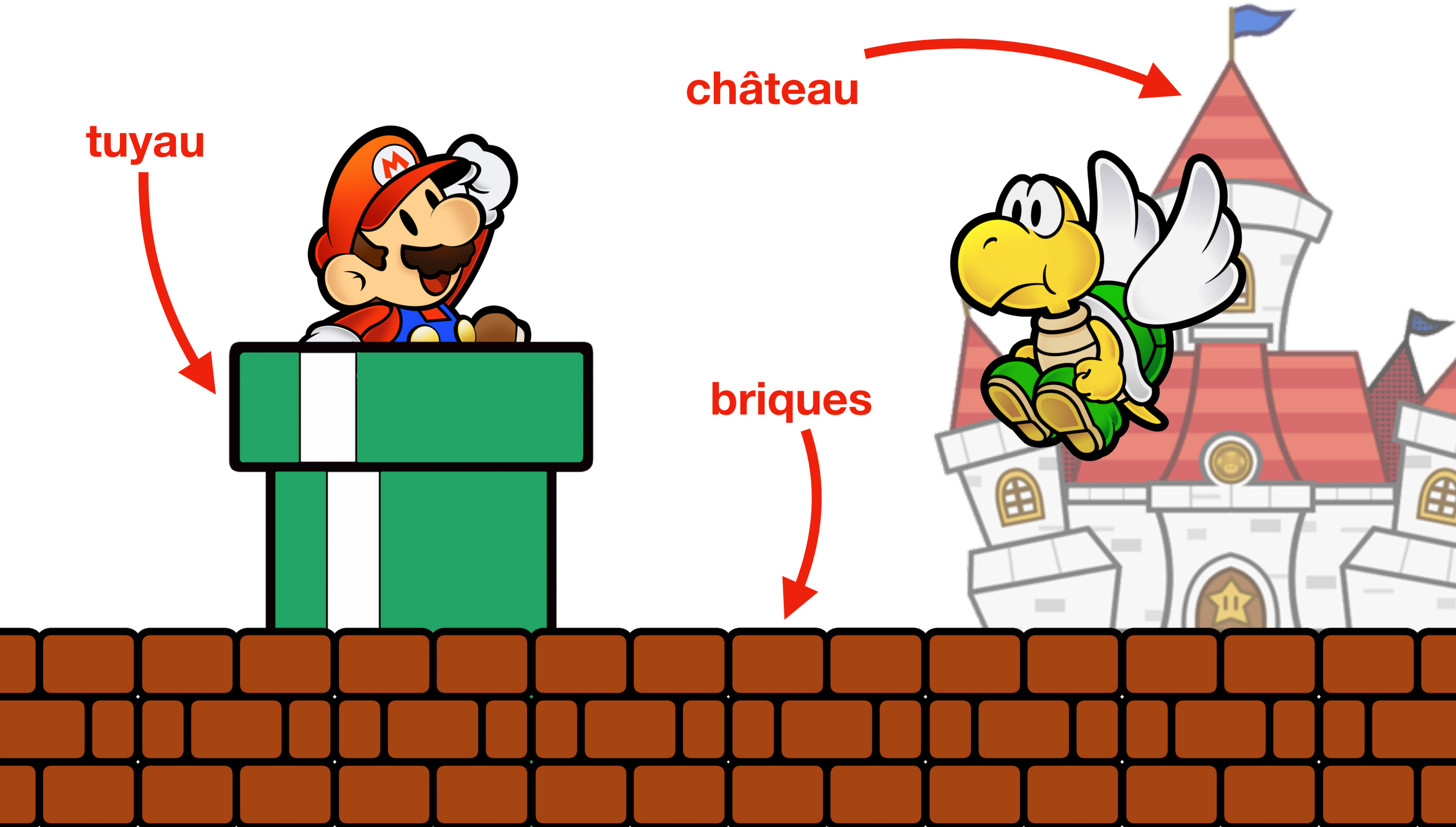
Algos de tri dans le jeux vidéo « Super Plombiers Italiens »



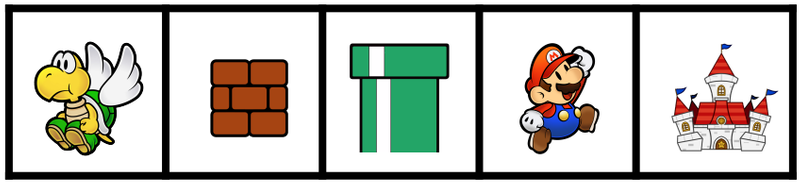
Algos de tri dans le jeux vidéo « Super Plombiers Italiens »



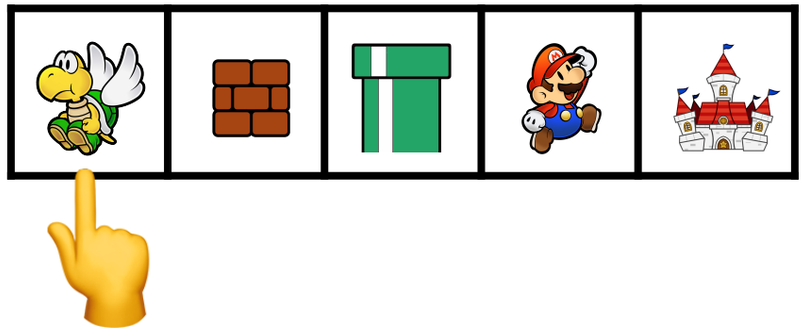
Algos de tri dans le jeux vidéo « Super Plombiers Italiens »



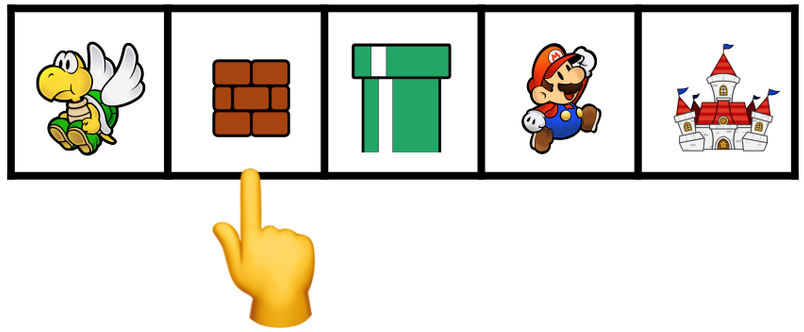
Affichage des objets dans le mauvais ordre



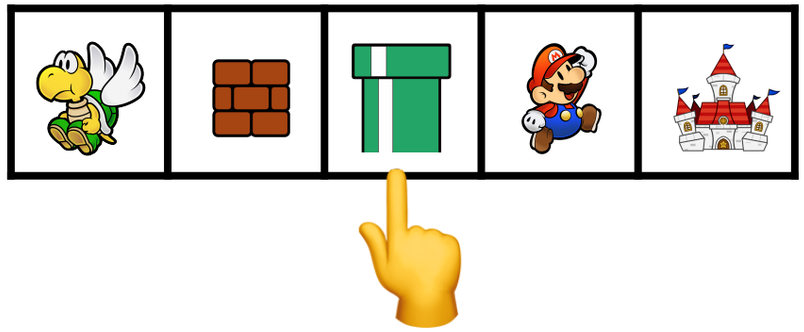
Affichage des objets dans le mauvais ordre

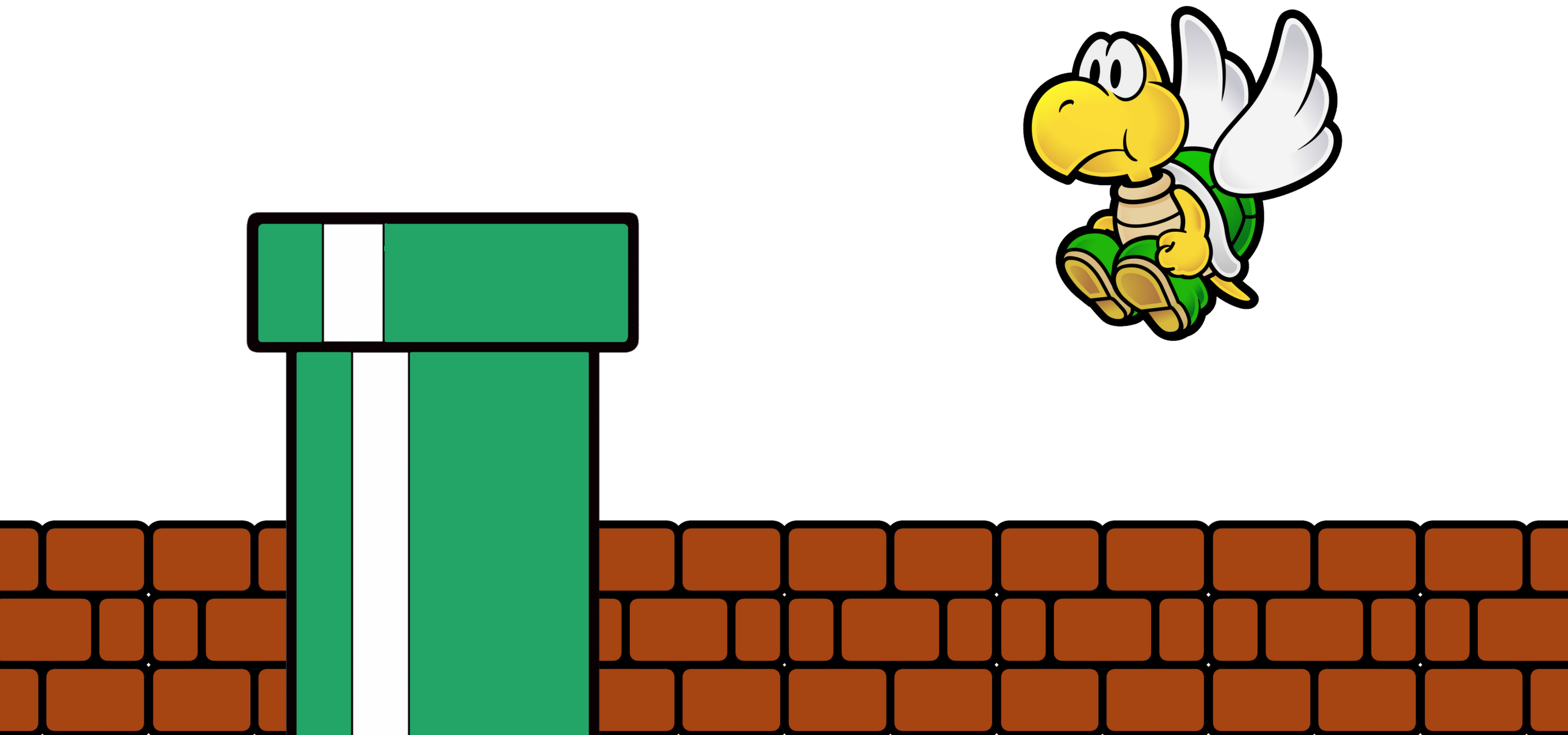


Affichage des objets dans le mauvais ordre

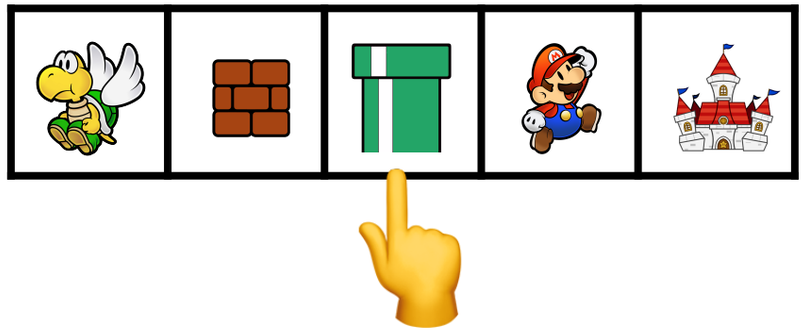


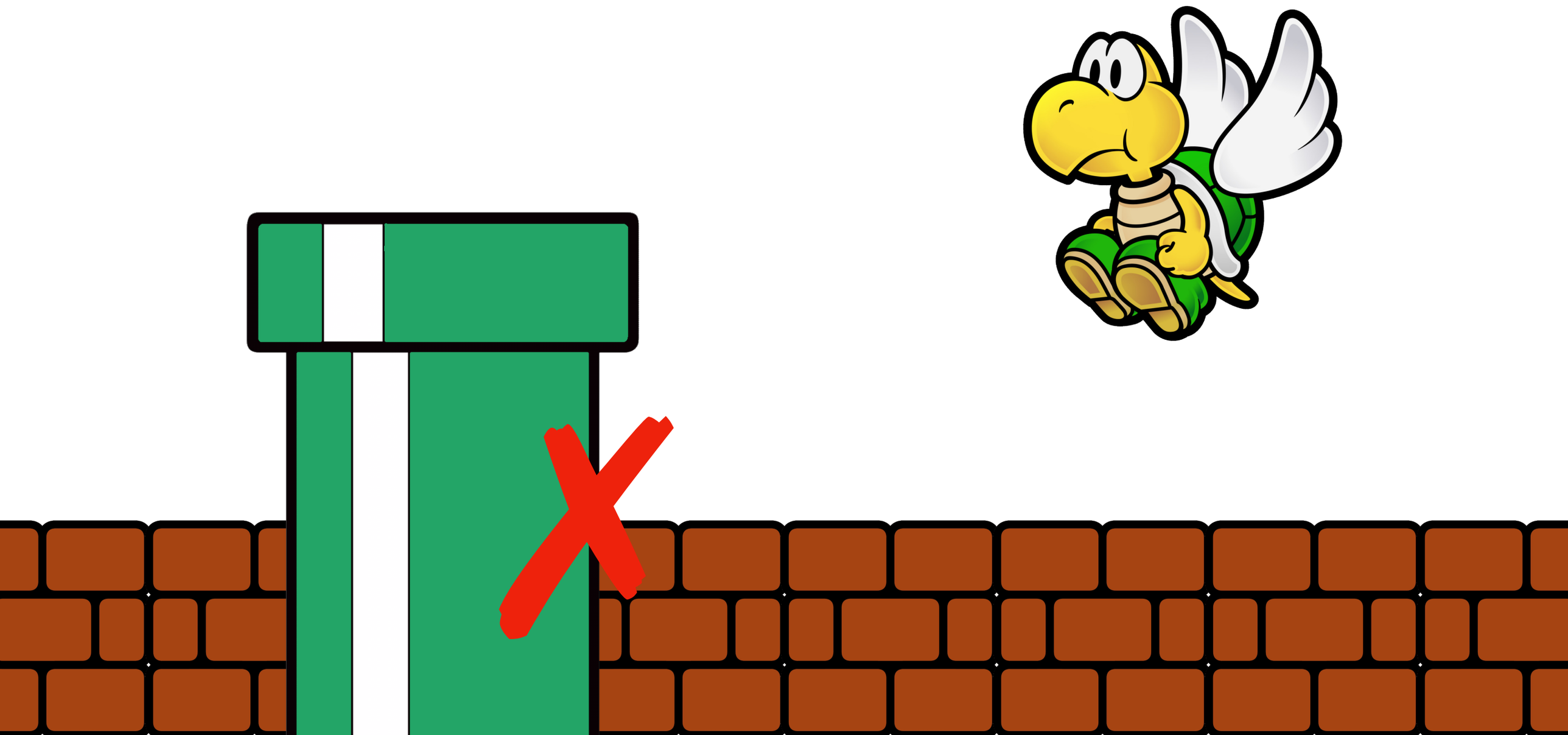
Affichage des objets dans le mauvais ordre



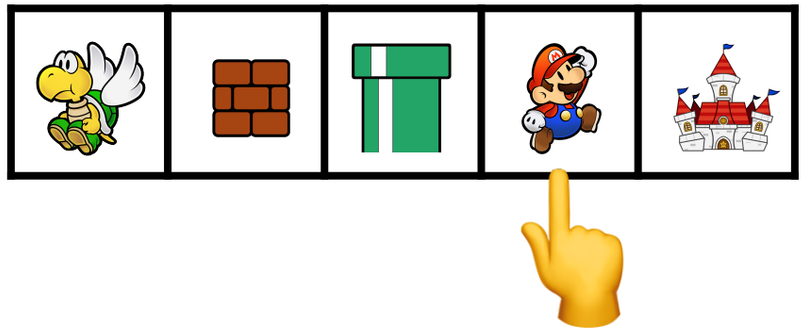


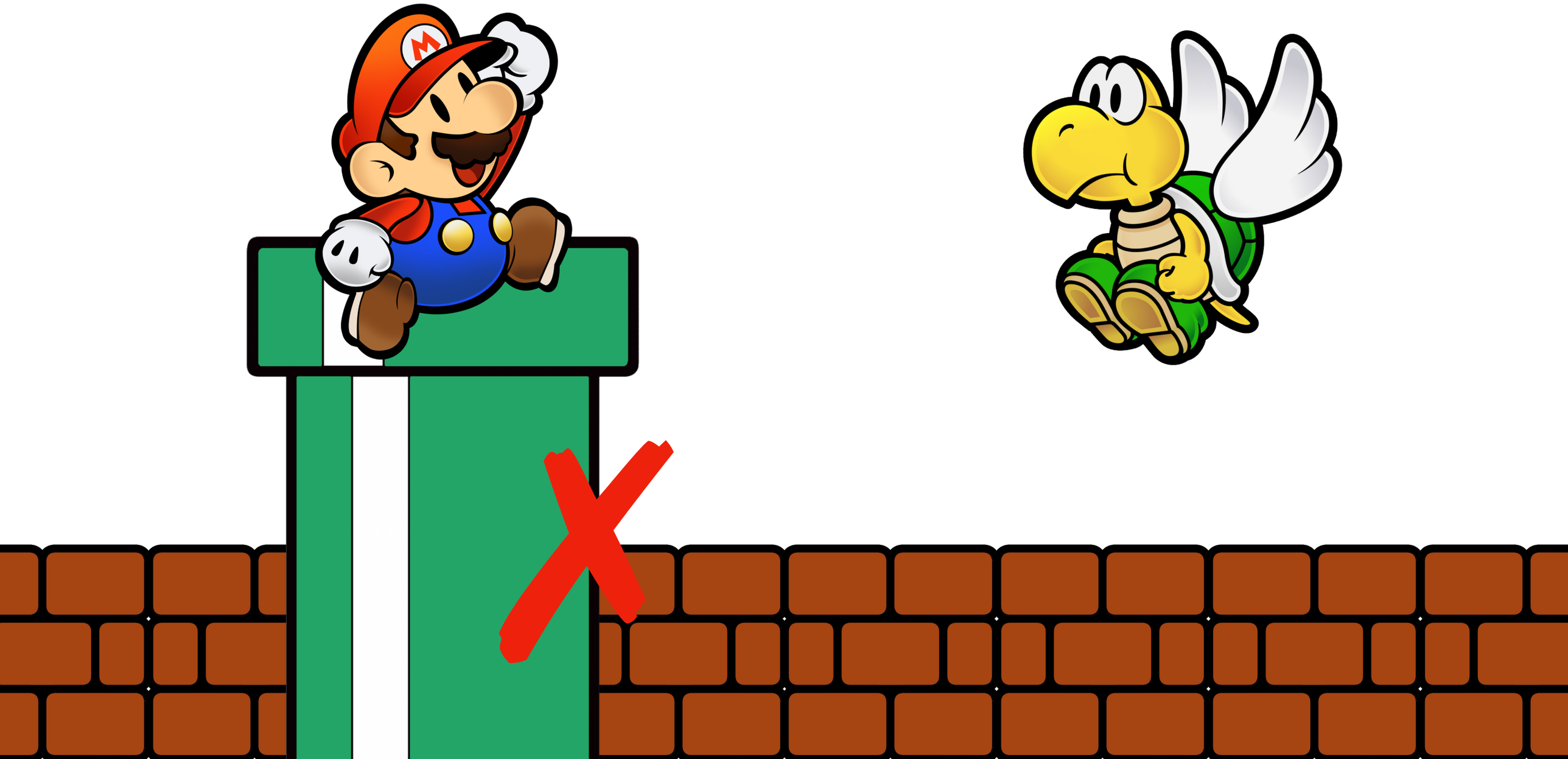
Affichage des objets dans le mauvais ordre



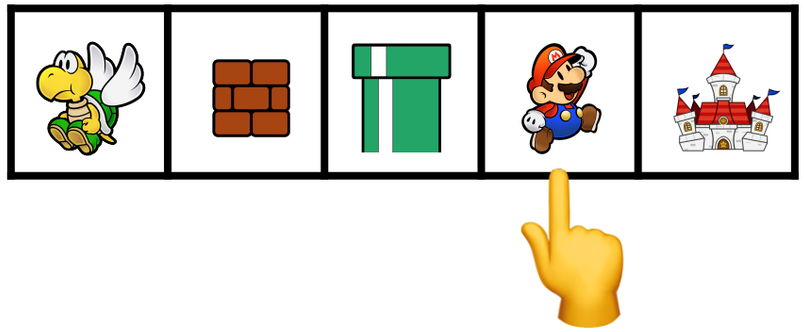


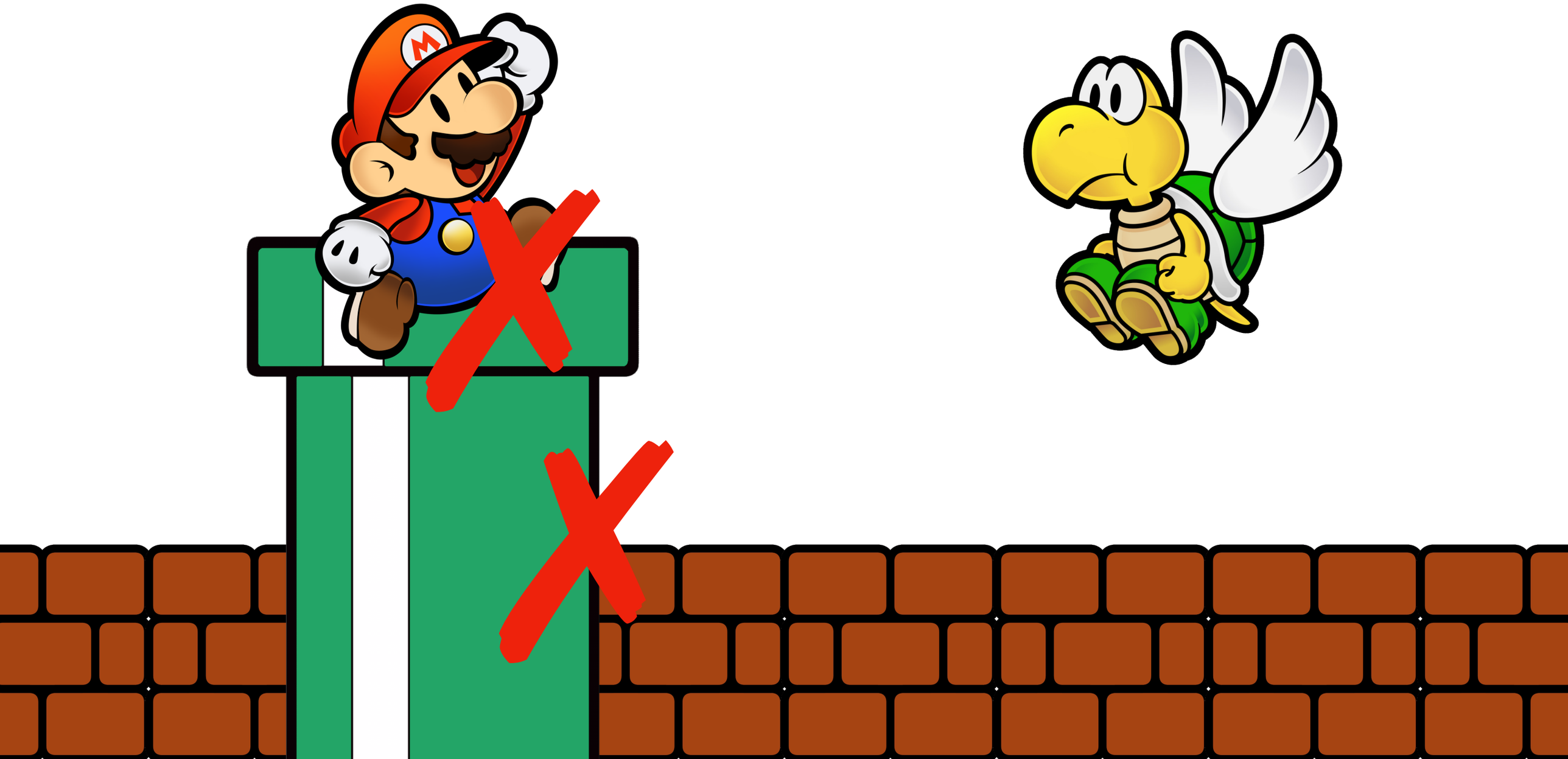
Affichage des objets dans le mauvais ordre



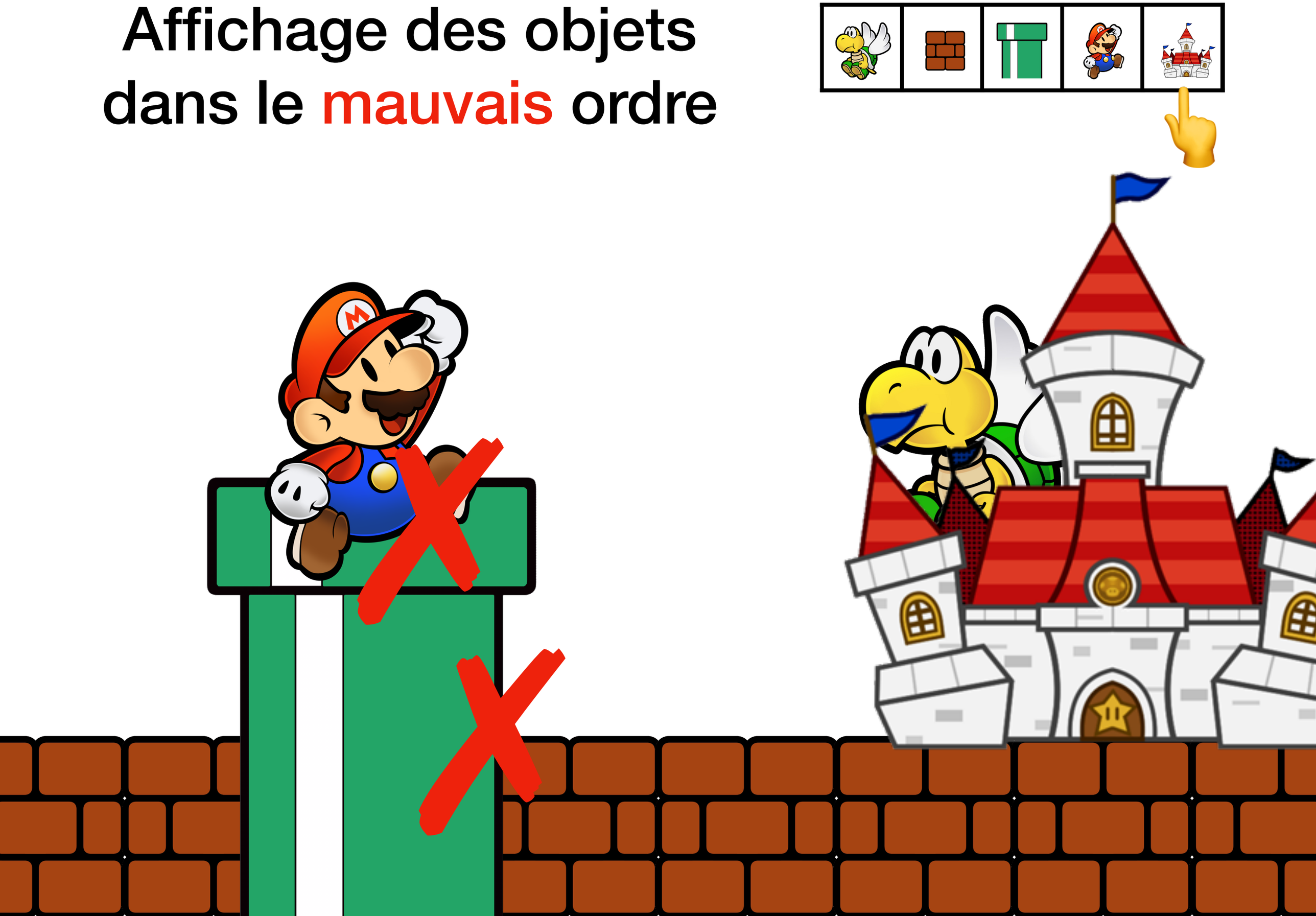


Affichage des objets dans le mauvais ordre

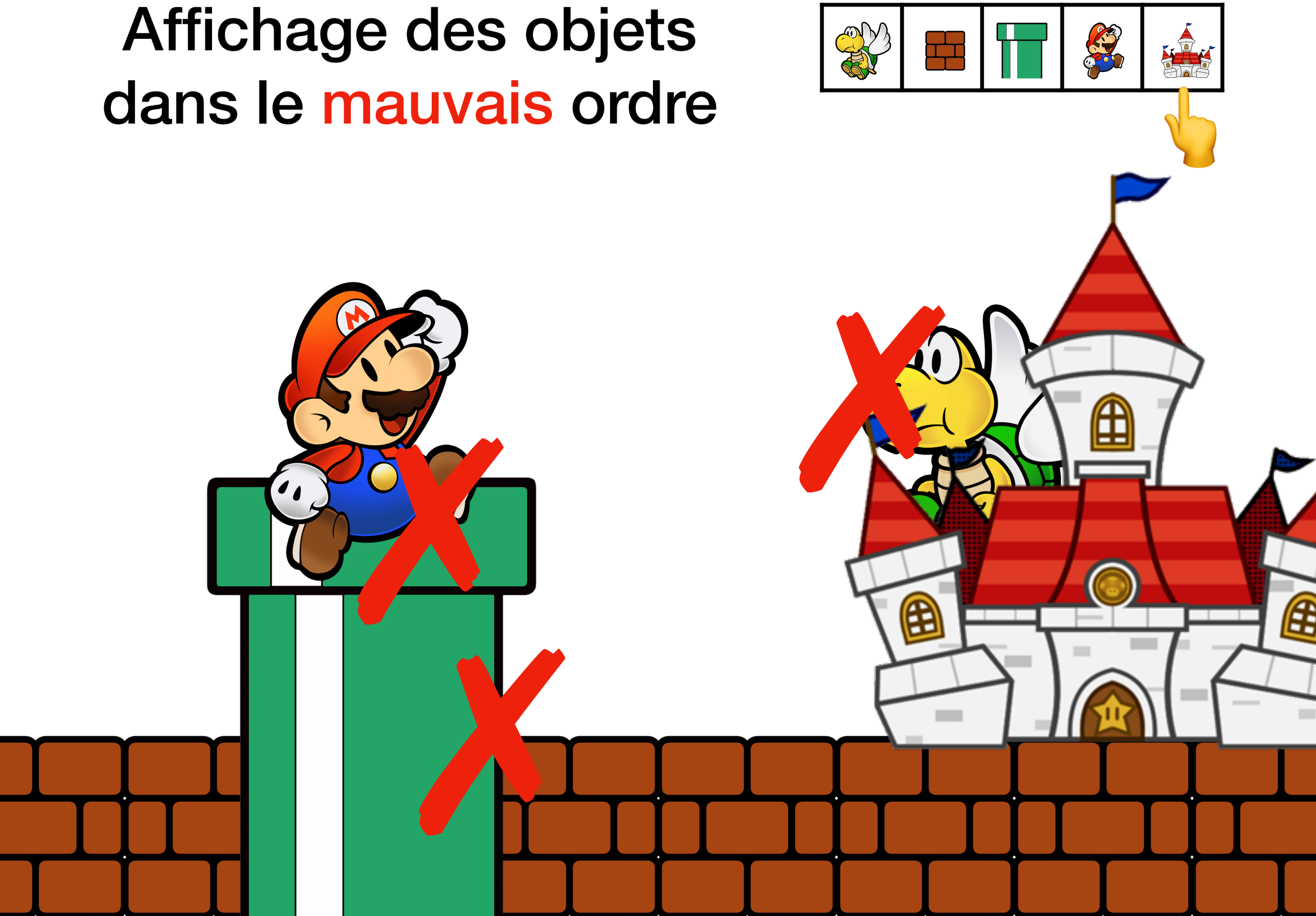




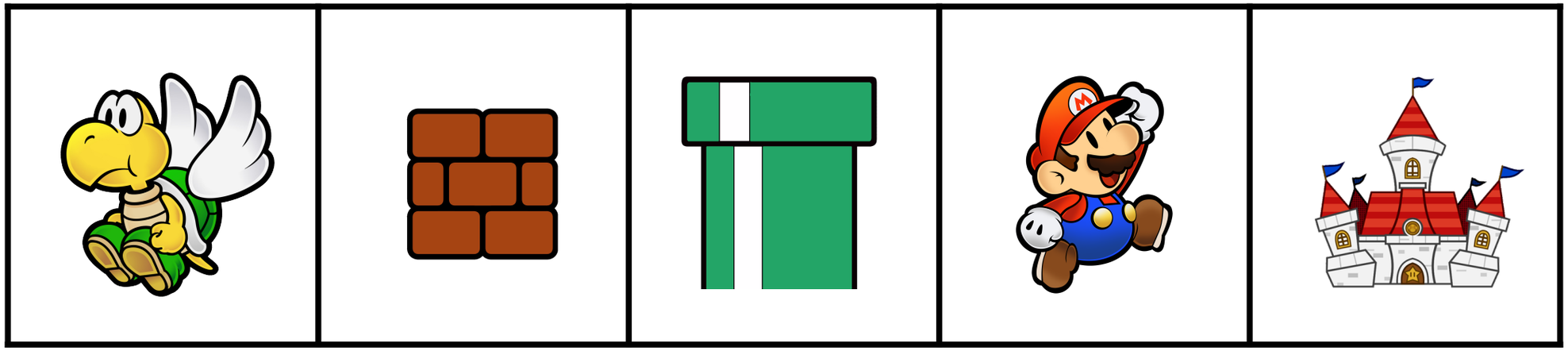
Affichage des objets dans le mauvais ordre



Affichage des objets dans le mauvais ordre

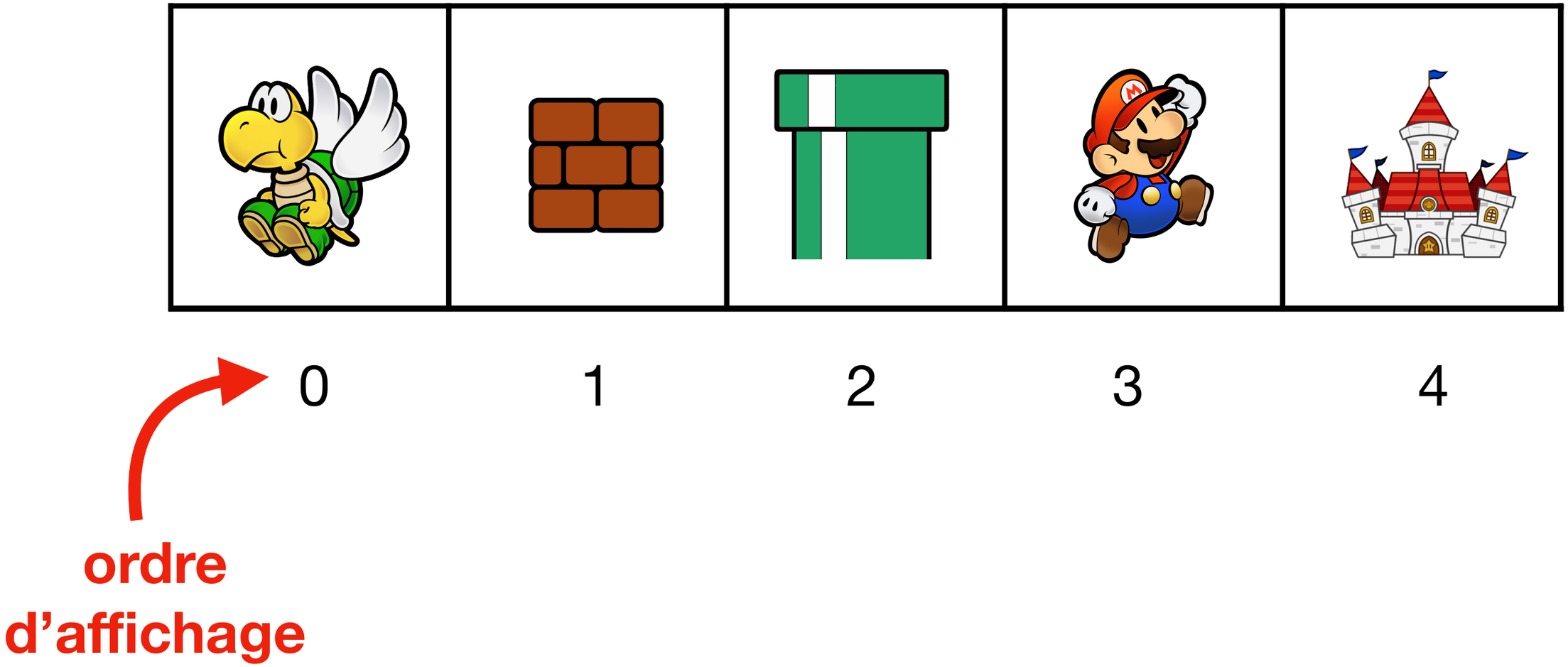


Affichage des objets dans le mauvais ordre

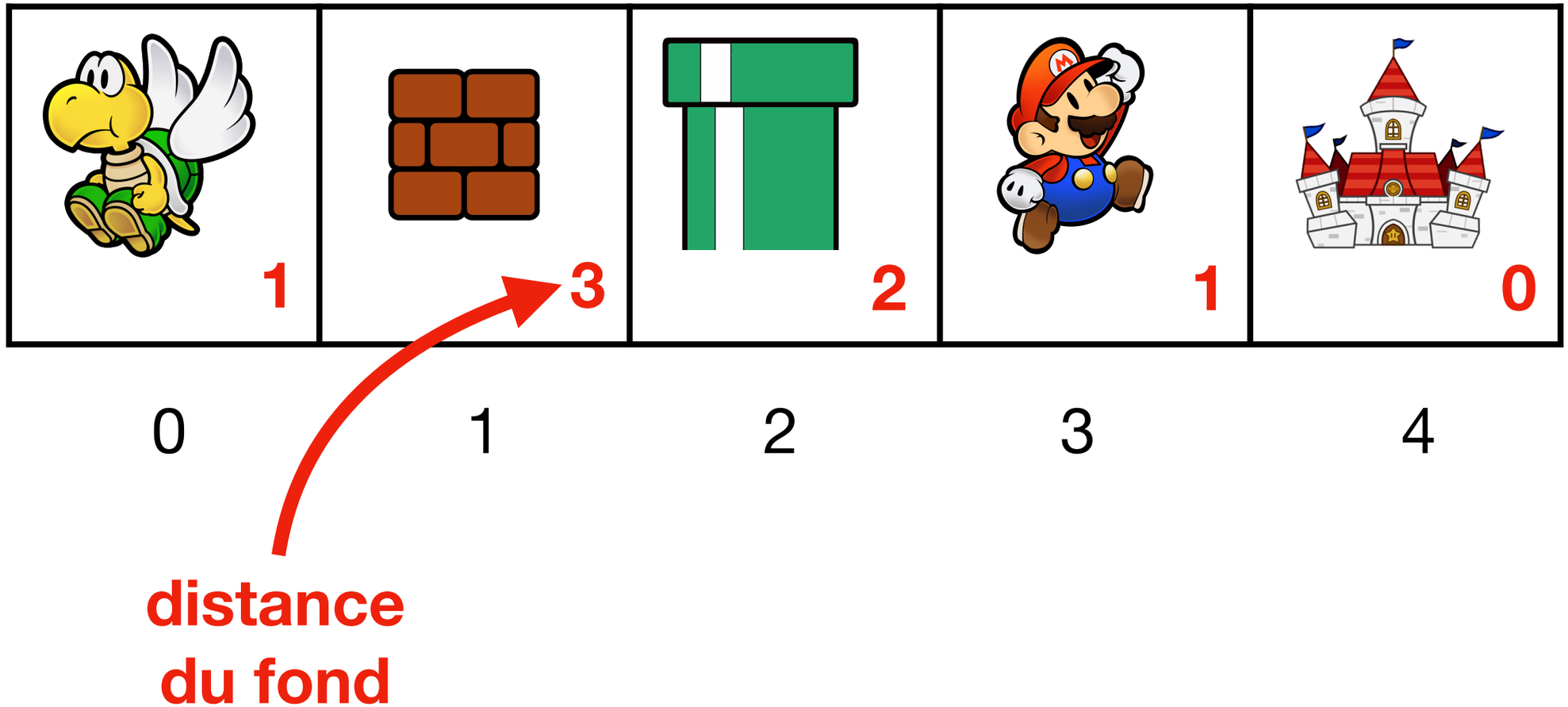


0 1 2 3 4

Affichage des objets dans le mauvais ordre

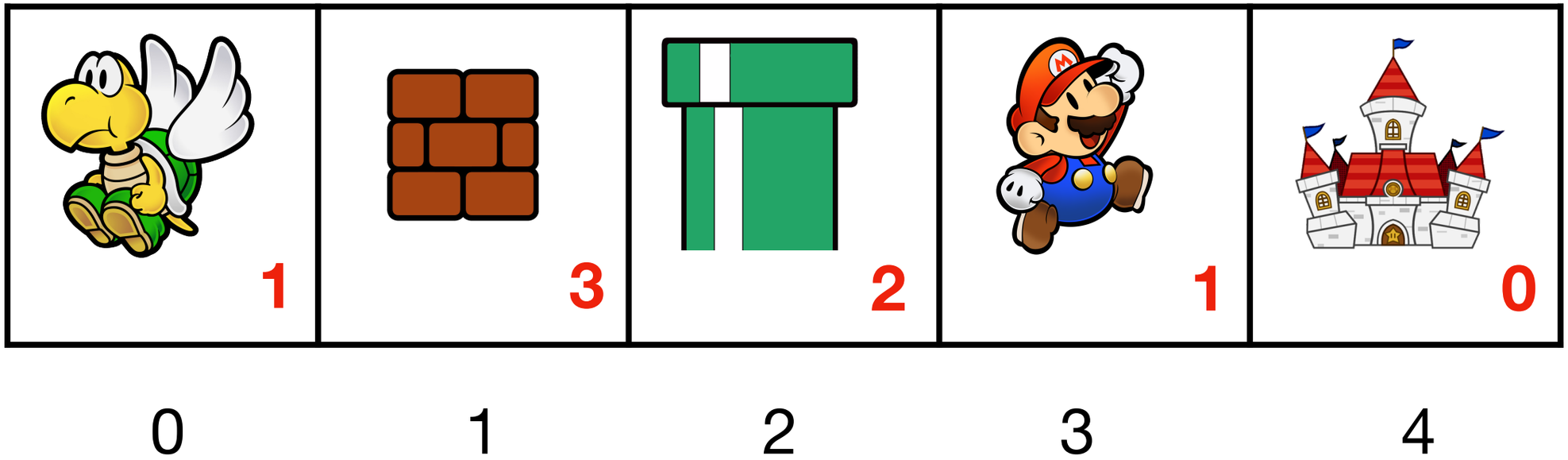


Affichage des objets dans le mauvais ordre

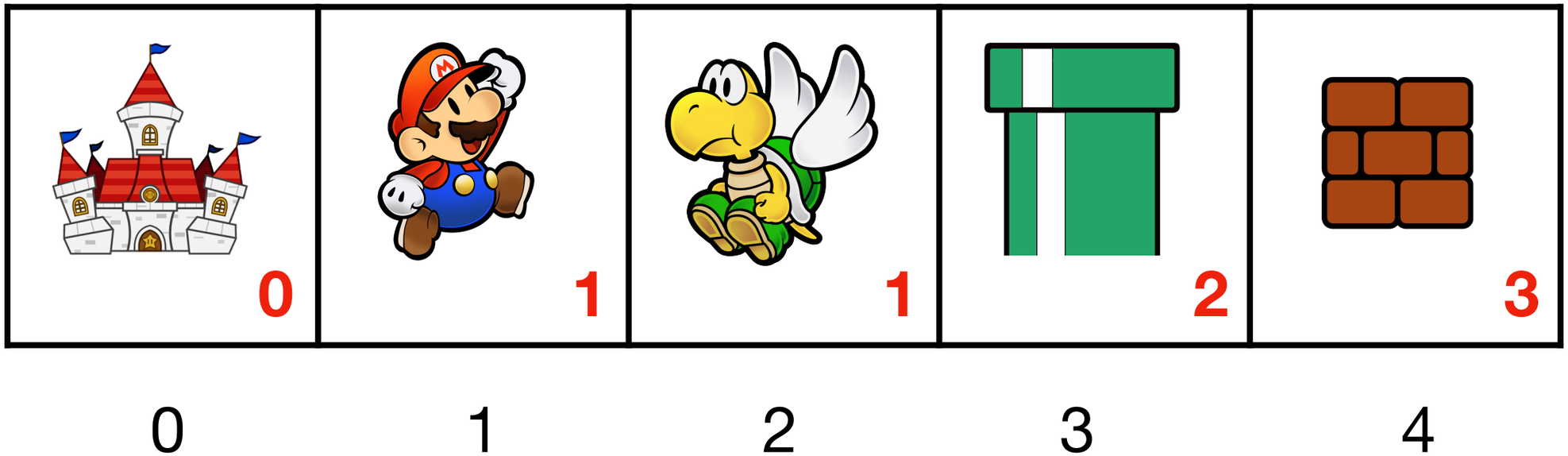


Tri ! ♻

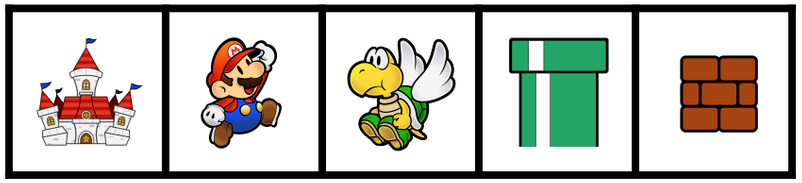
Affichage des objets dans le mauvais ordre



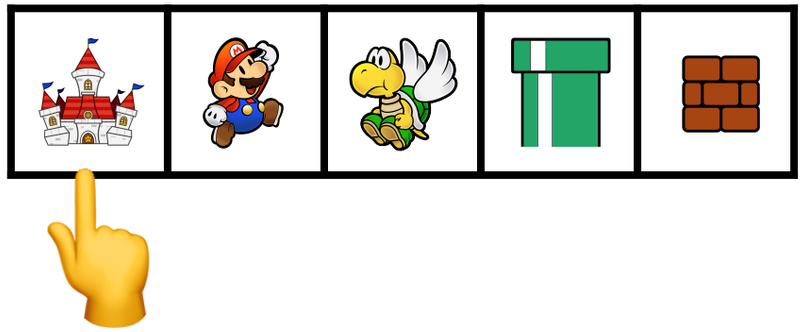
Affichage des objets dans le bon ordre



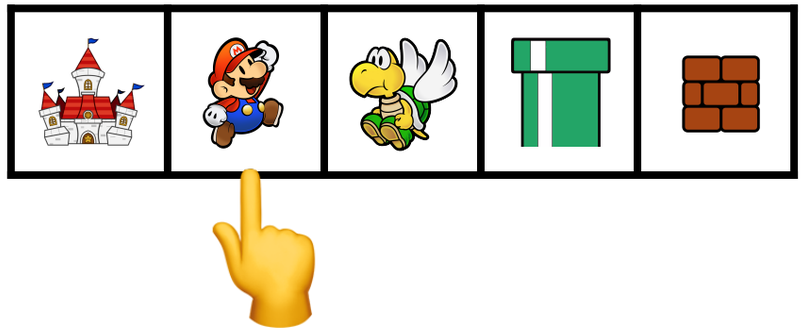
Affichage des objets dans le bon ordre



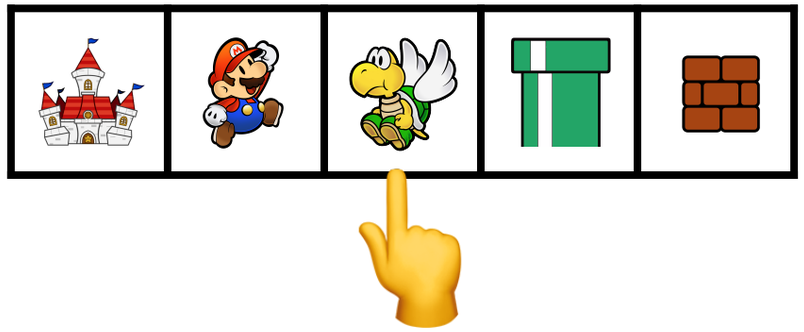
Affichage des objets dans le bon ordre



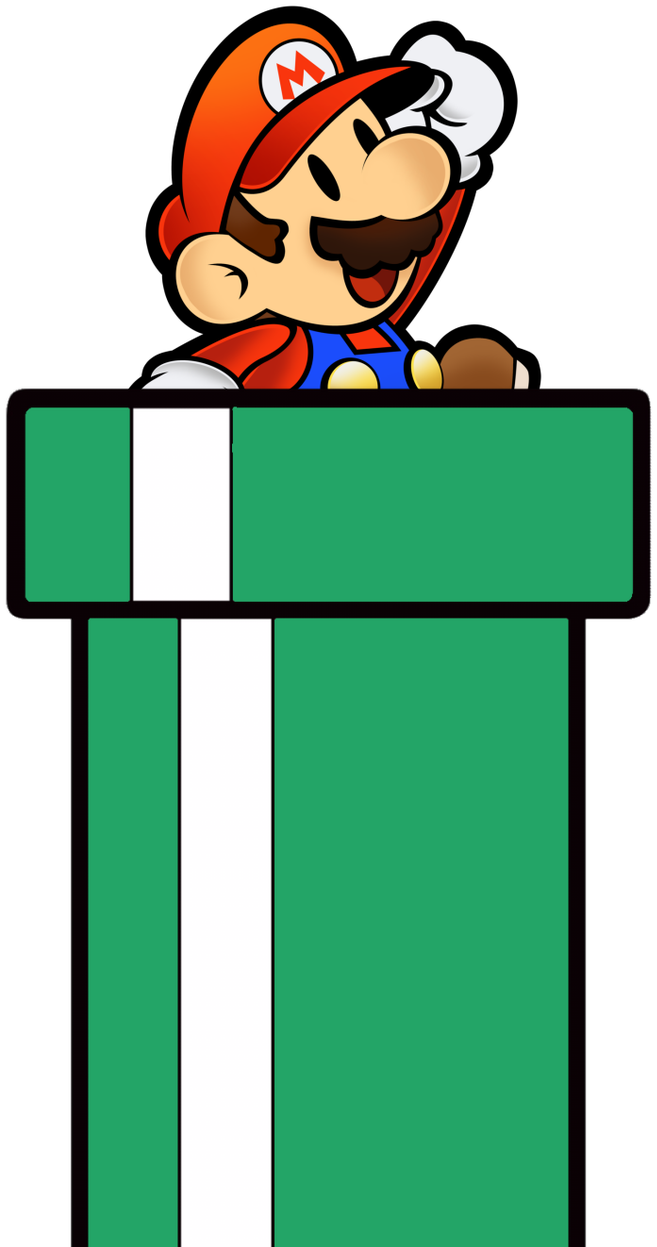
Affichage des objets dans le bon ordre



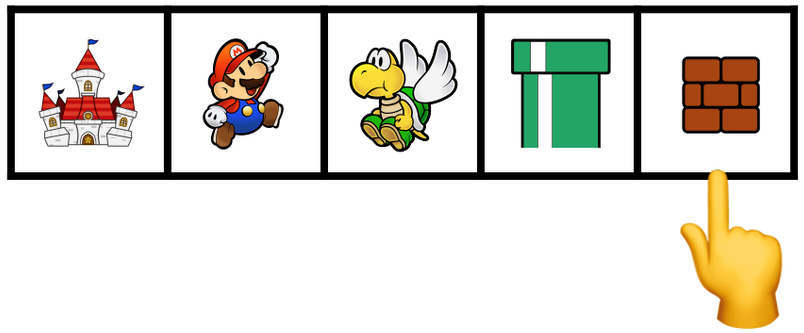
Affichage des objets dans le bon ordre

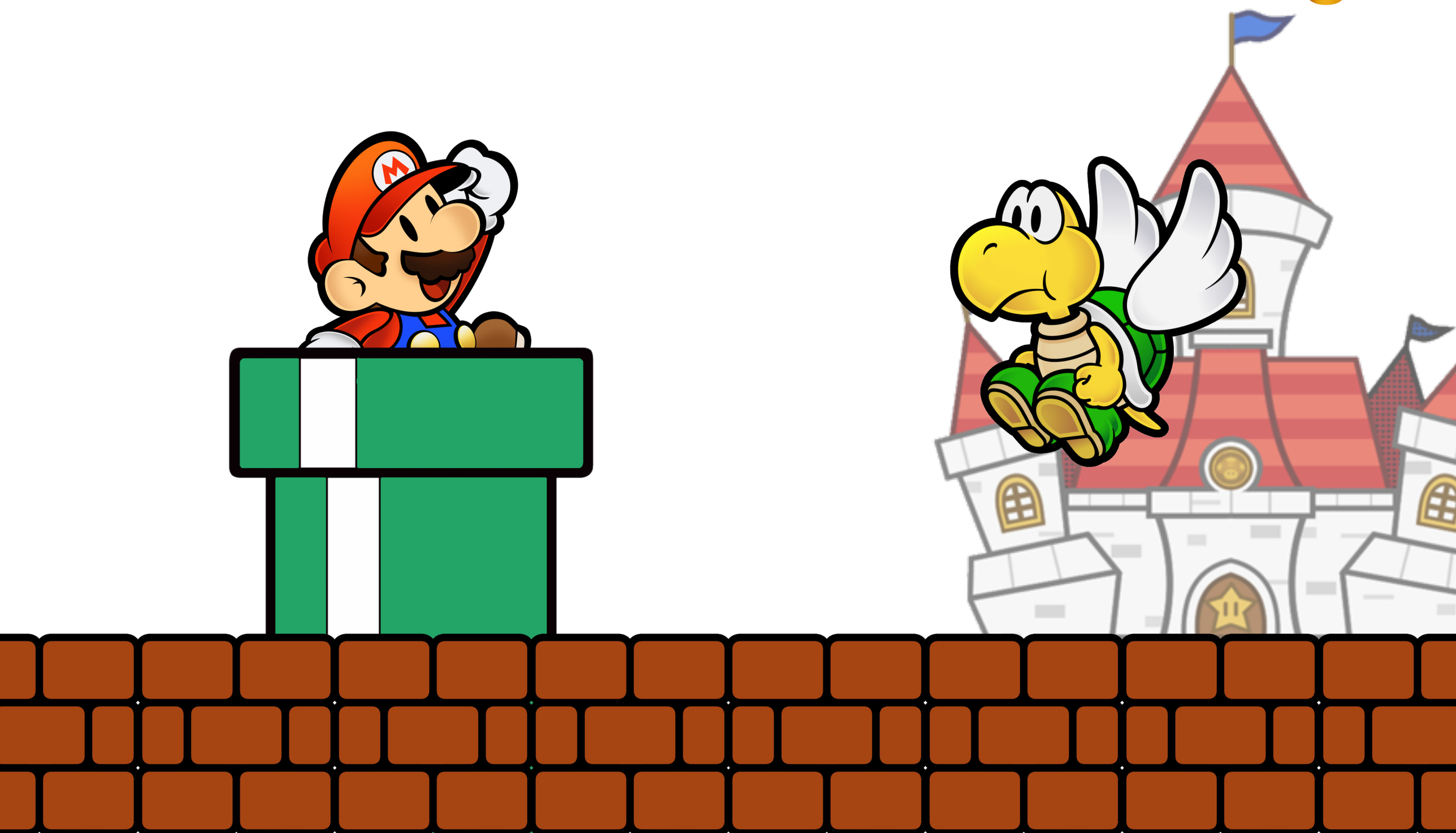


Affichage des objets dans le bon ordre

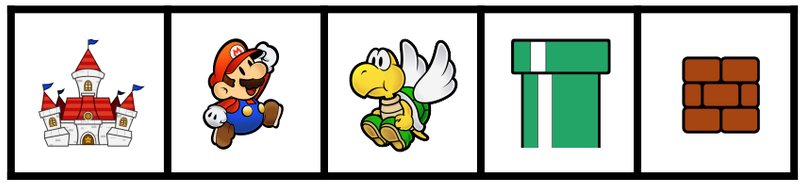


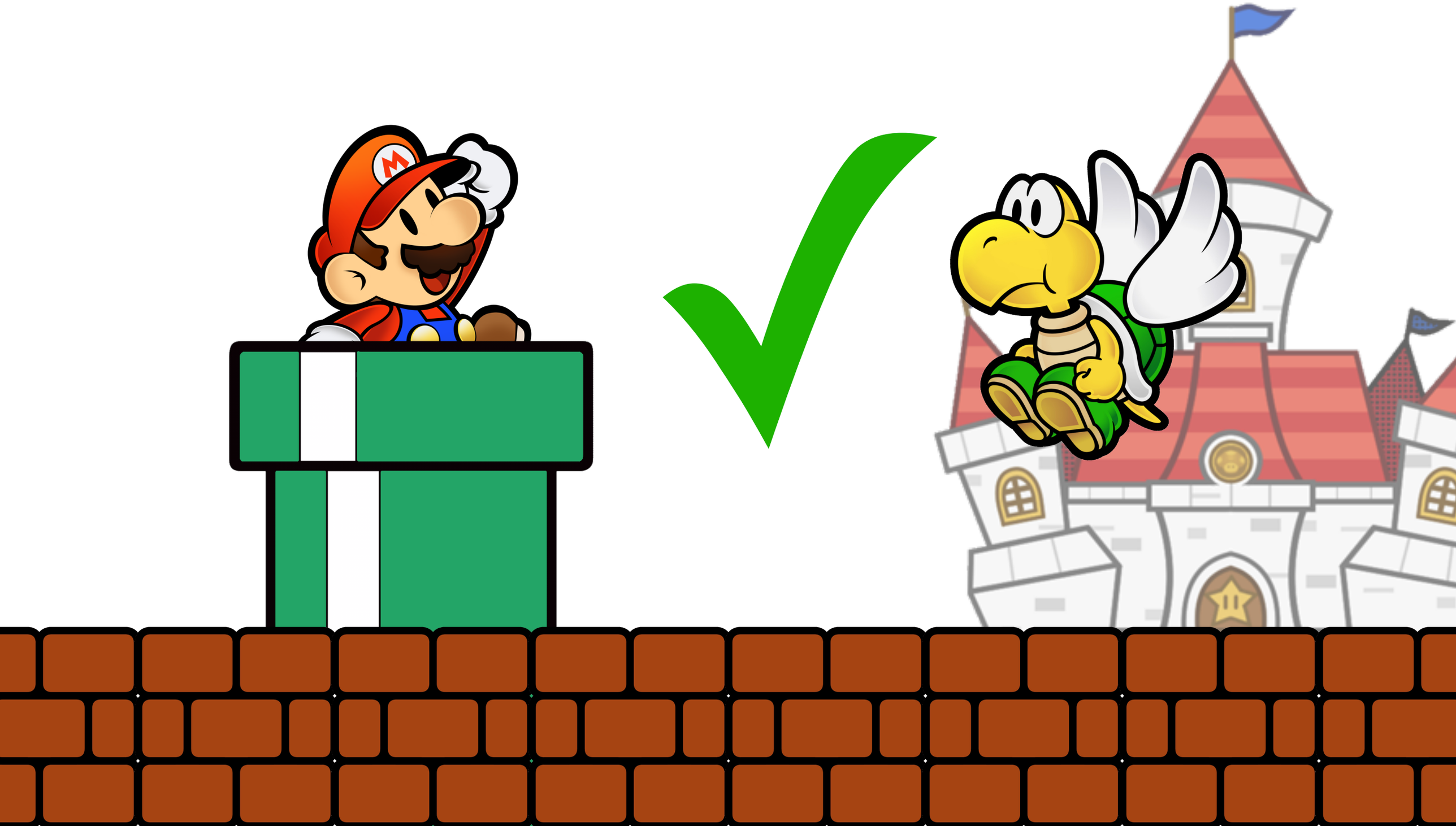
Affichage des objets dans le bon ordre





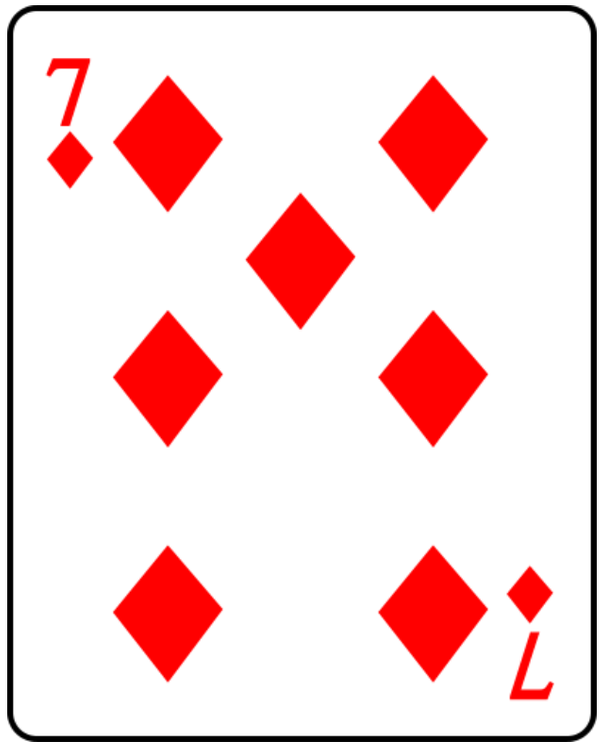
Affichage des objets dans le bon ordre



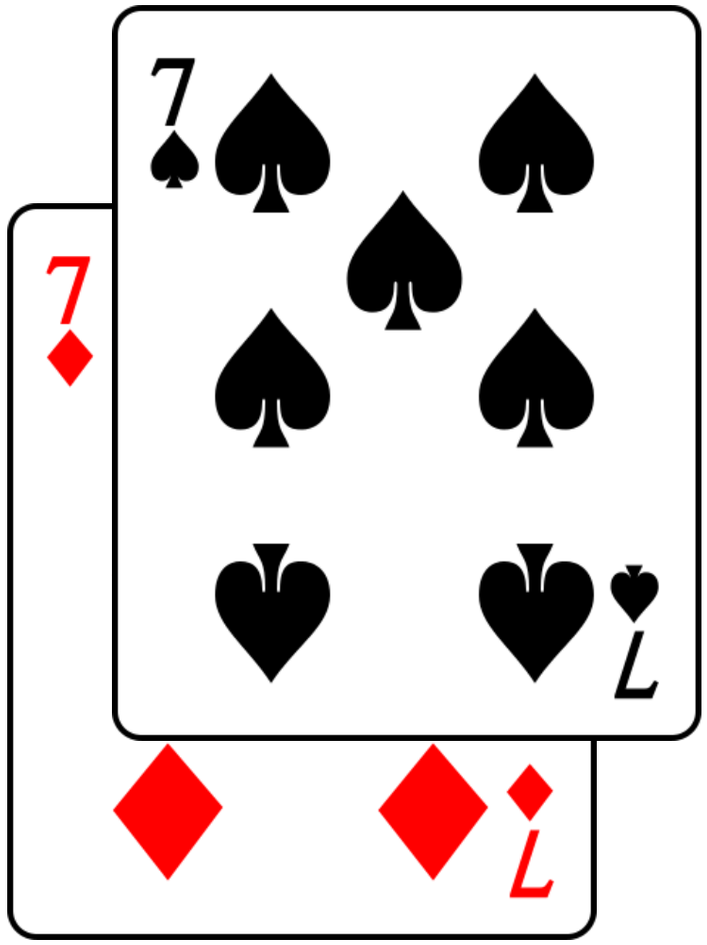


Tri par insertion

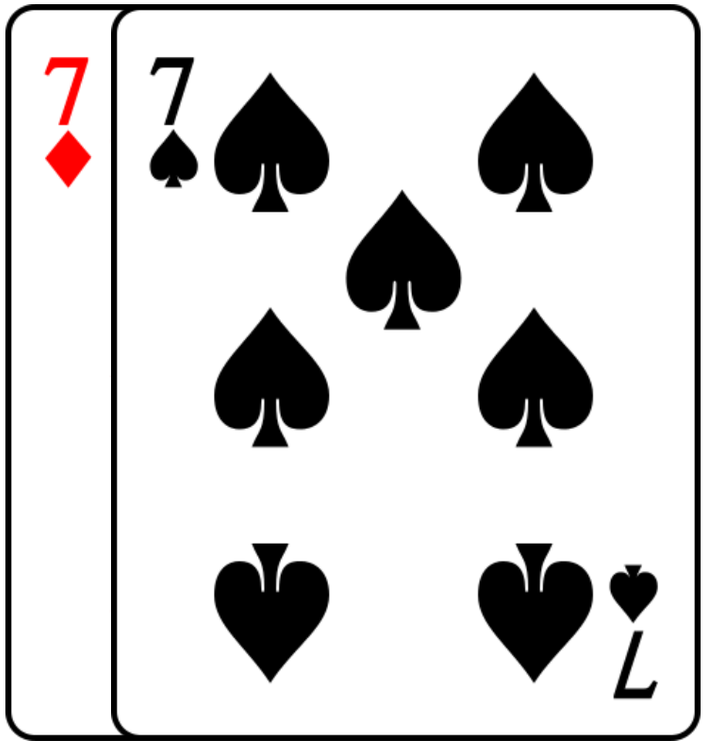
Tri par insertion



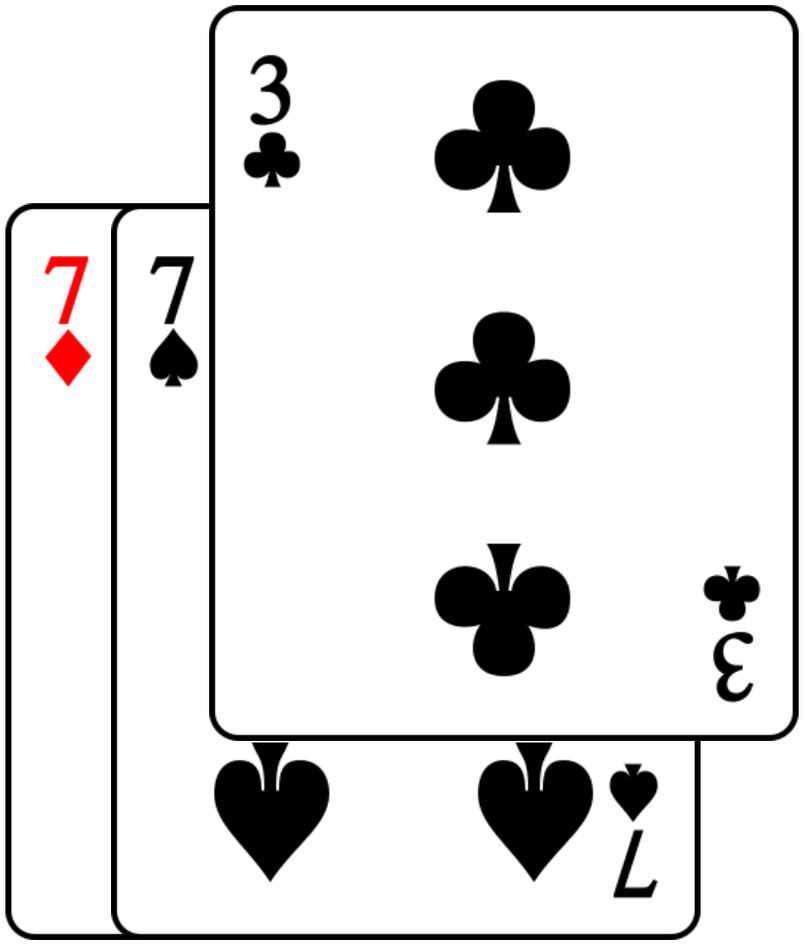
Tri par insertion



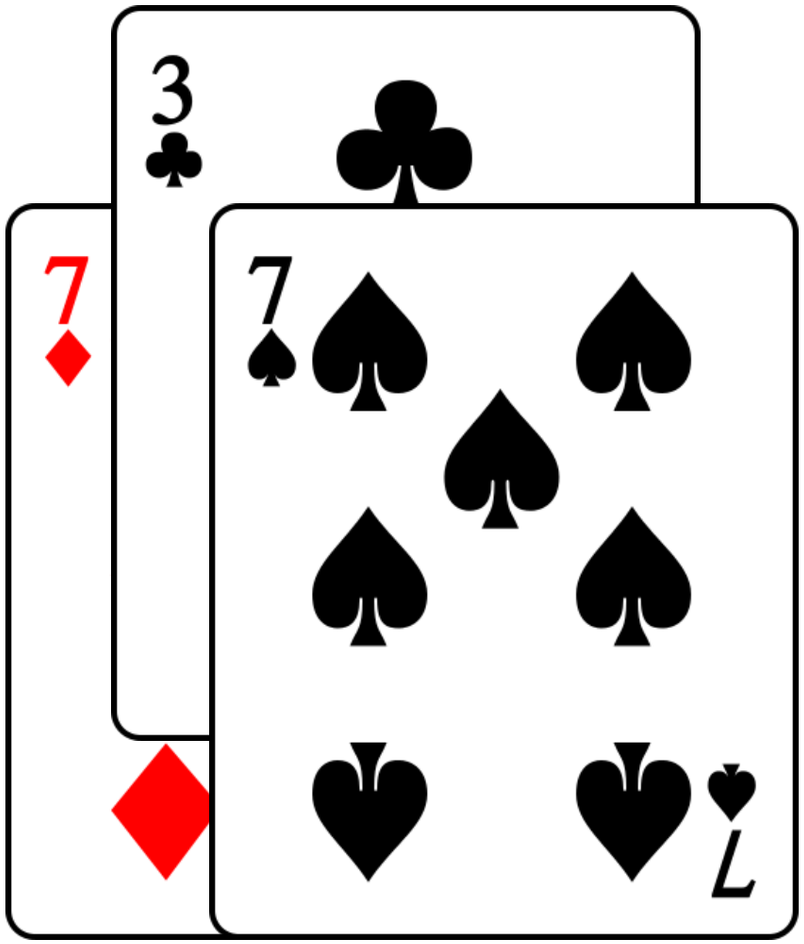
Tri par insertion



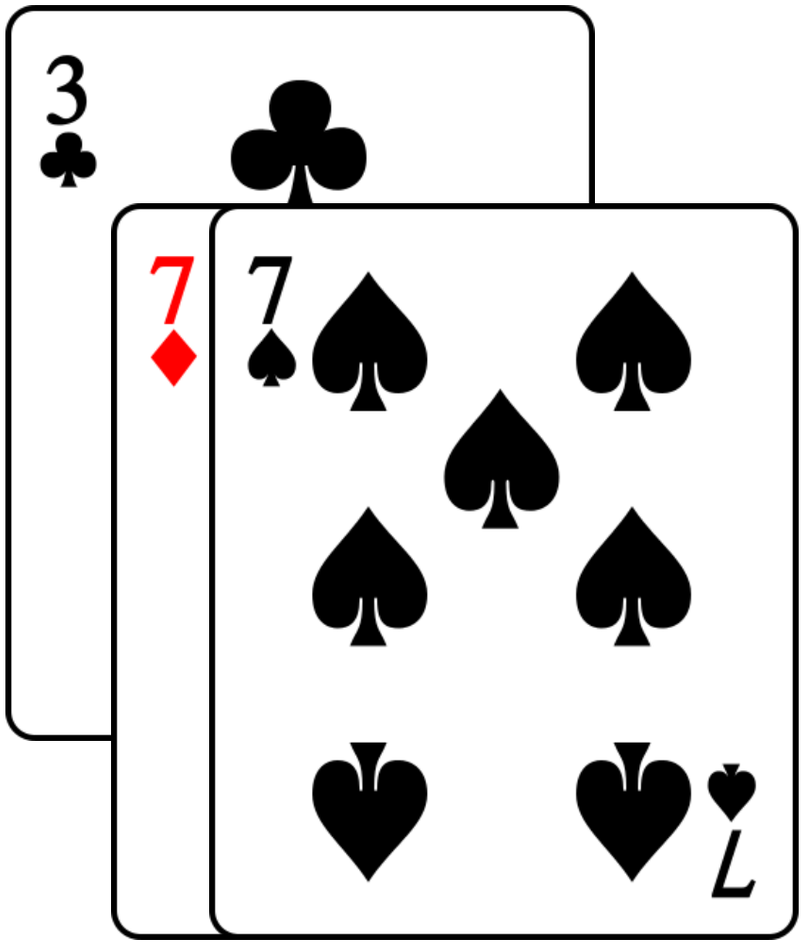
Tri par insertion



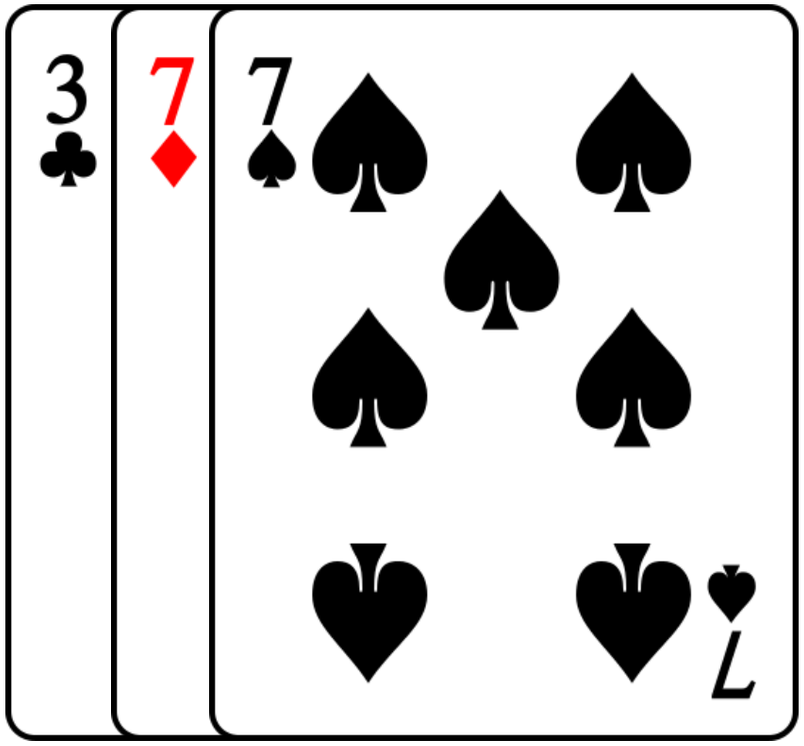
Tri par insertion



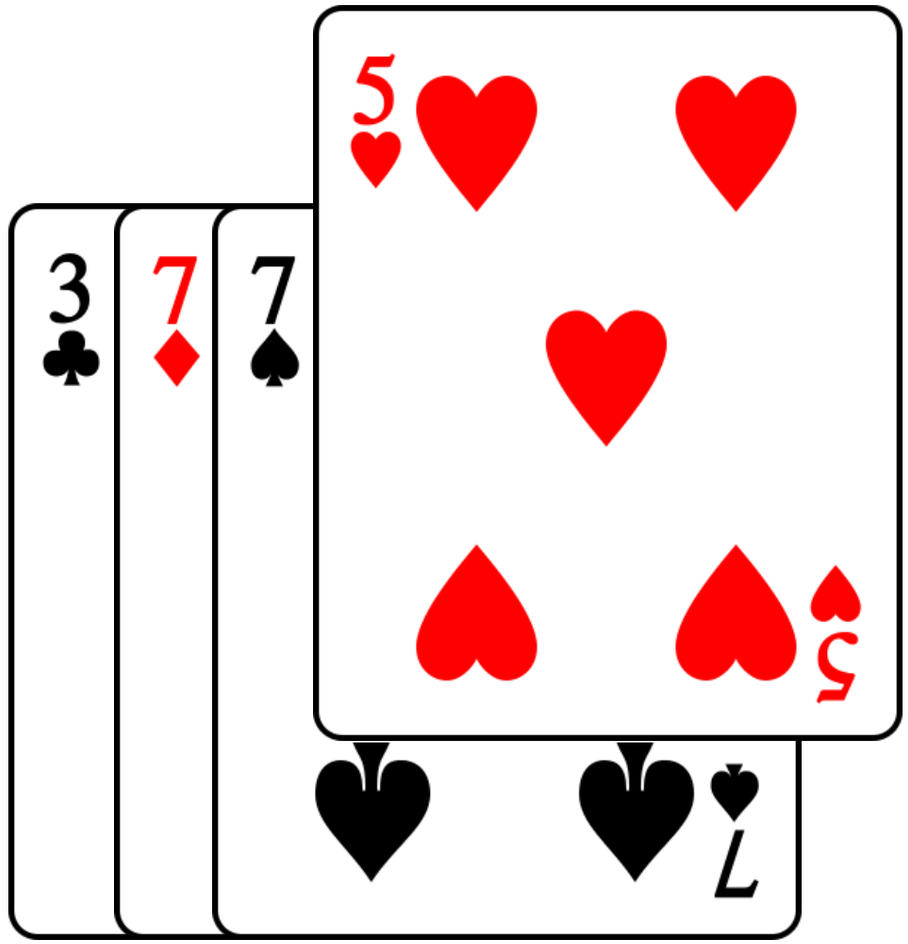
Tri par insertion



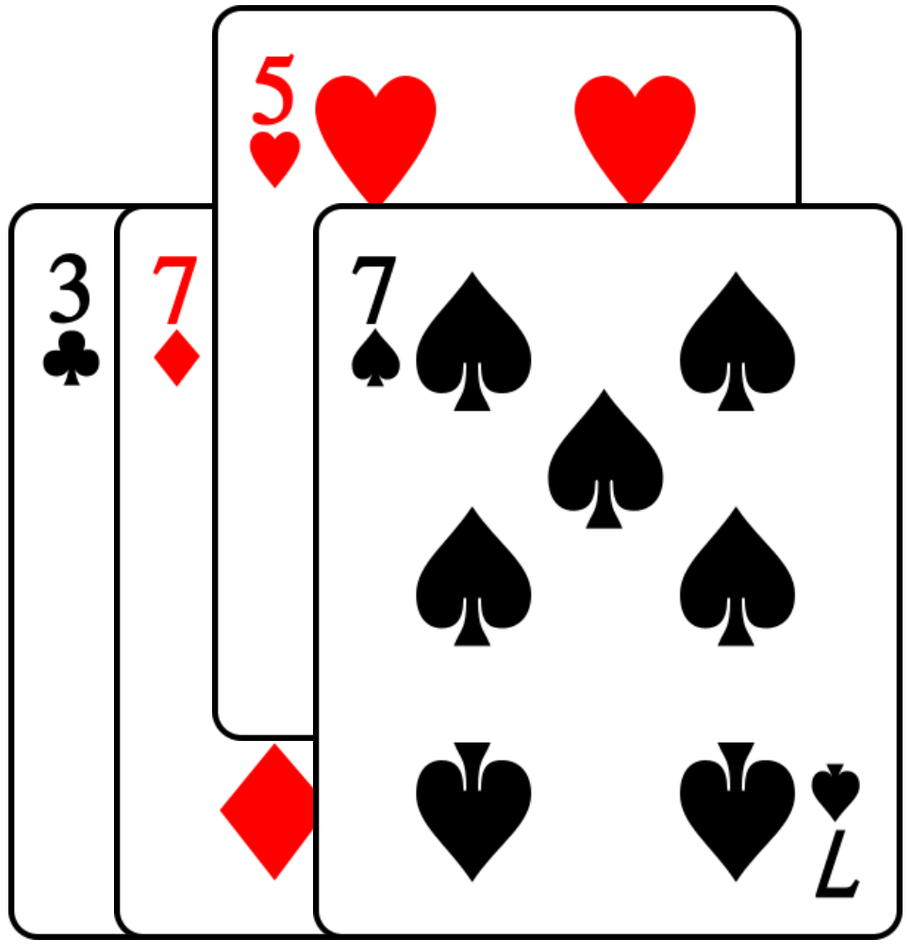
Tri par insertion



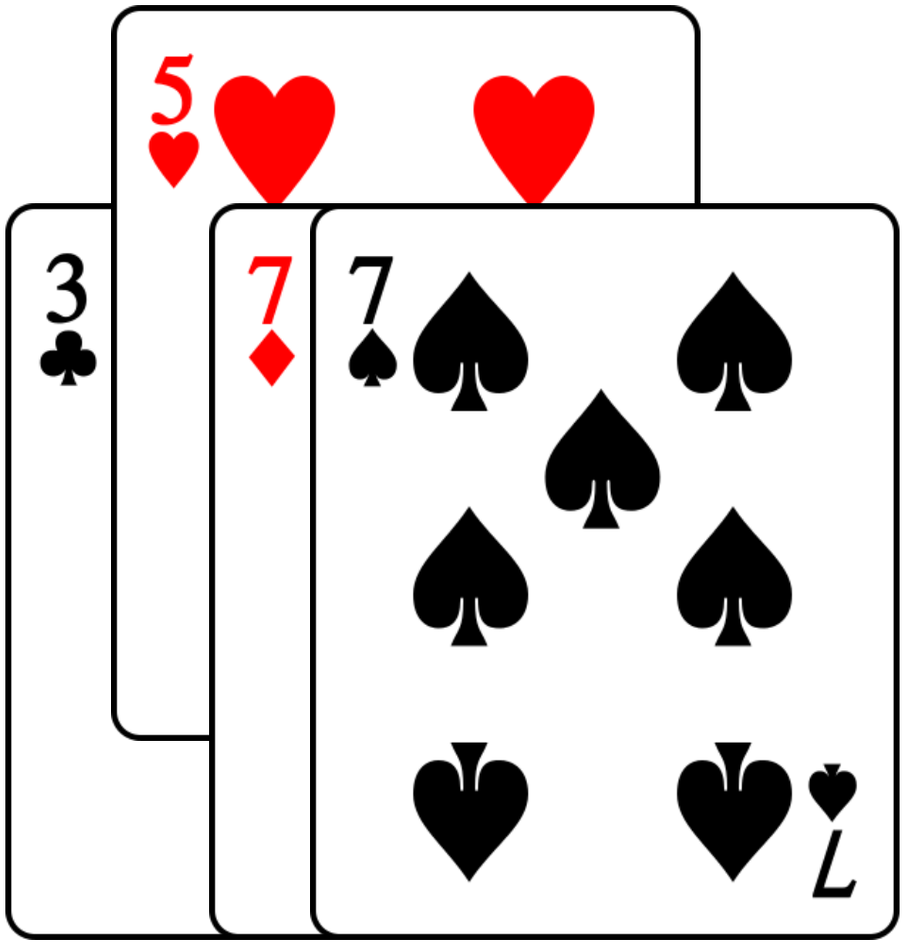
Tri par insertion



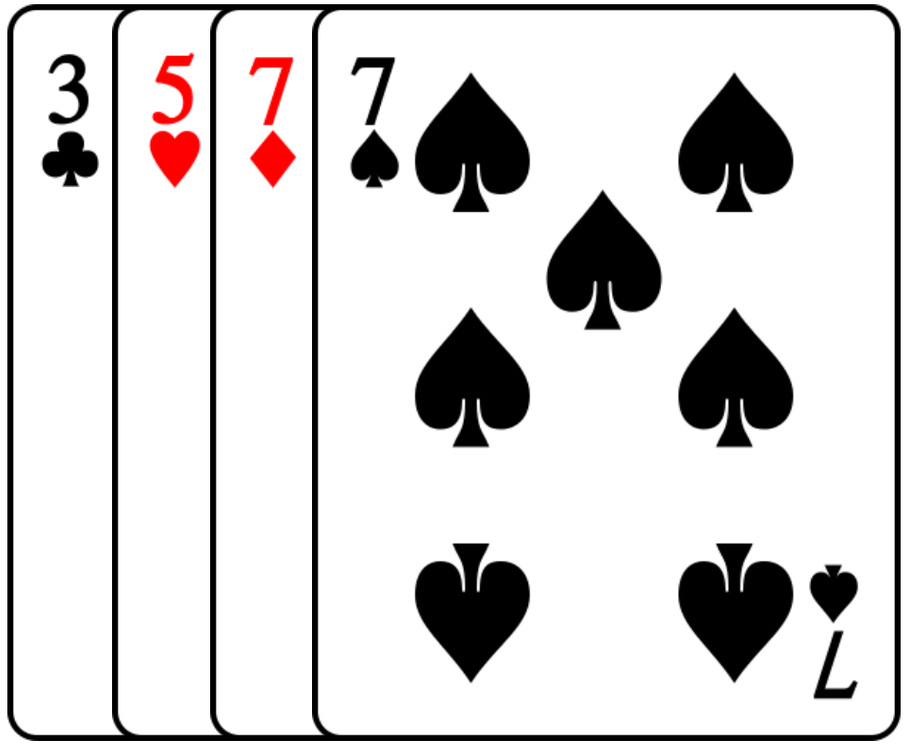
Tri par insertion



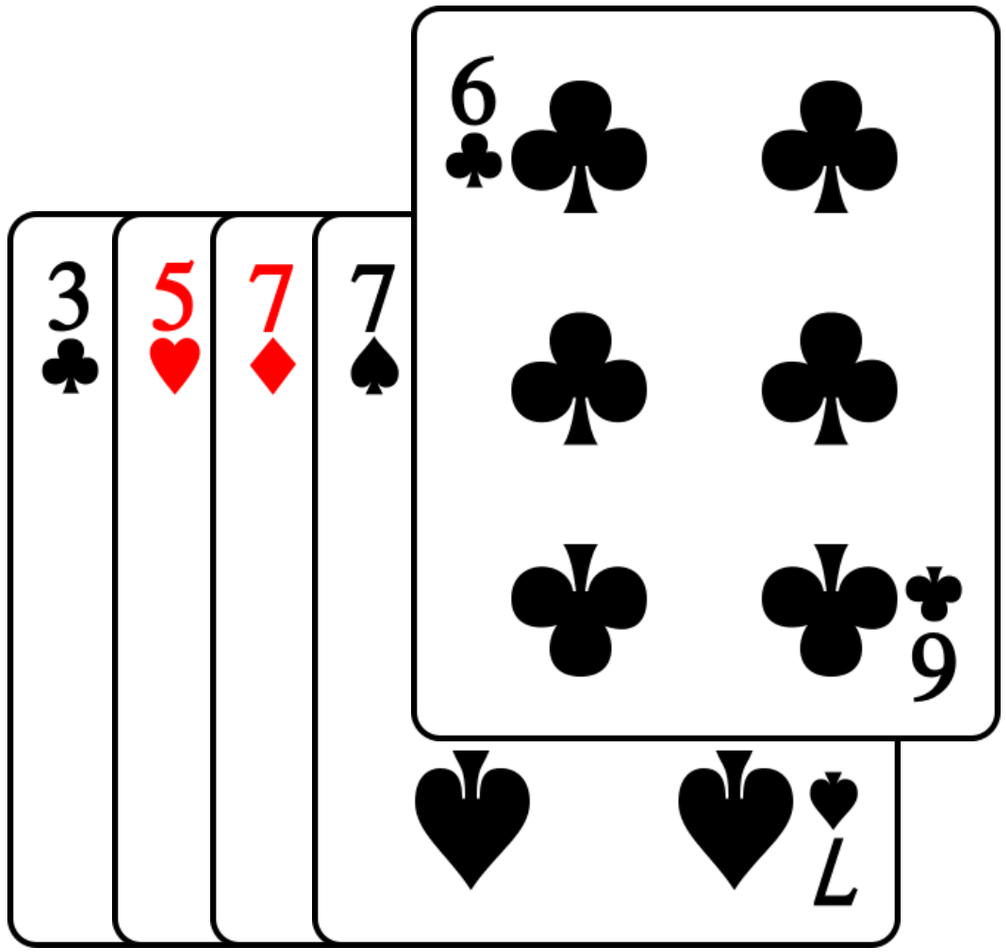
Tri par insertion



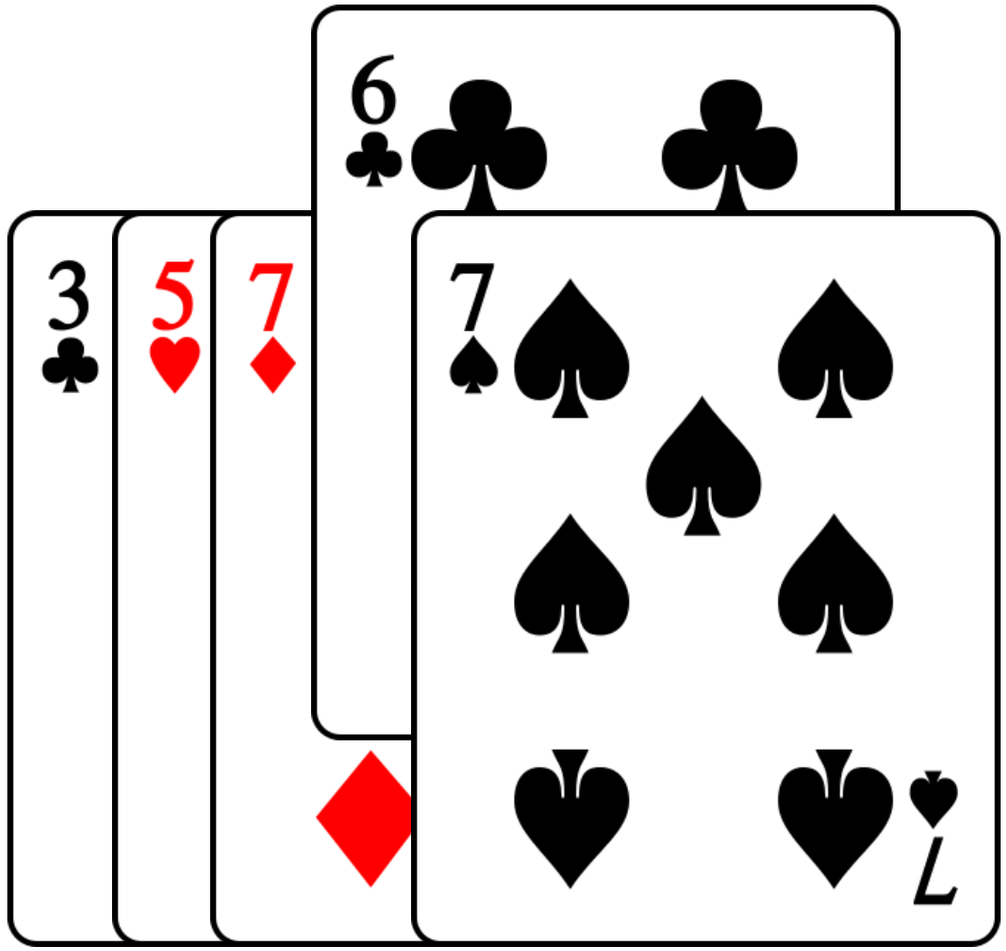
Tri par insertion



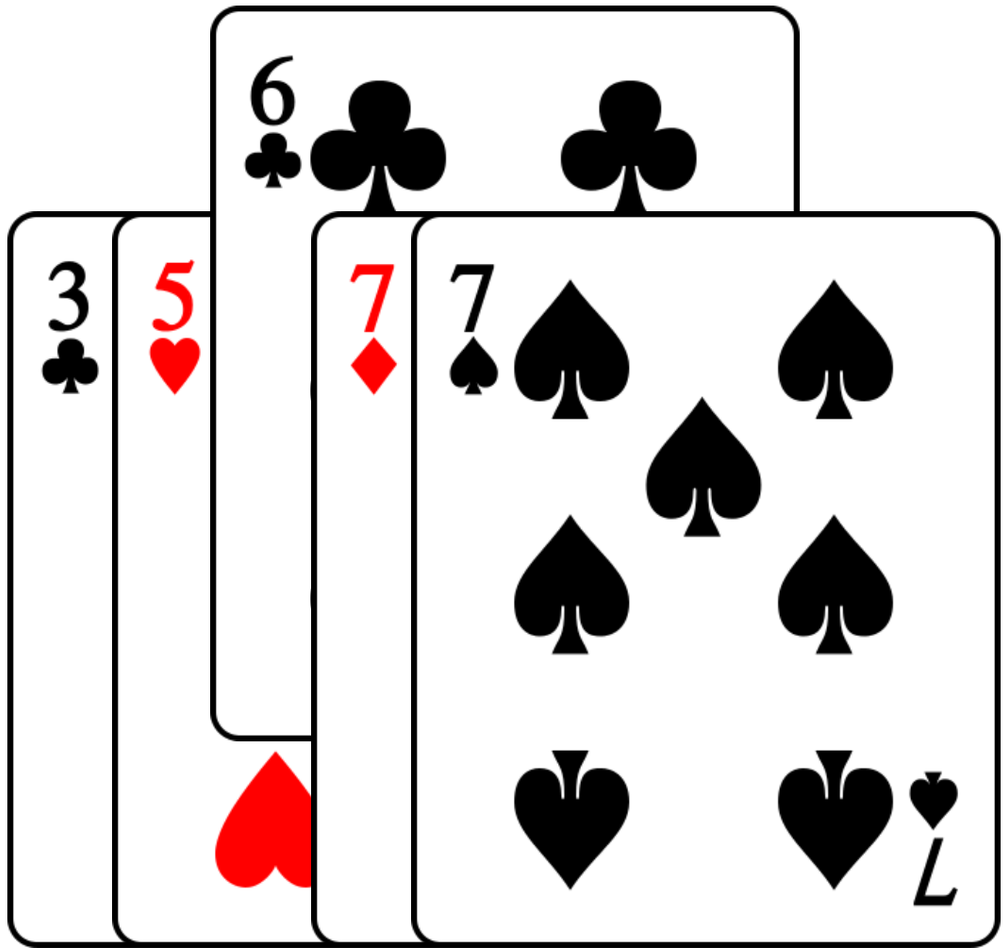
Tri par insertion



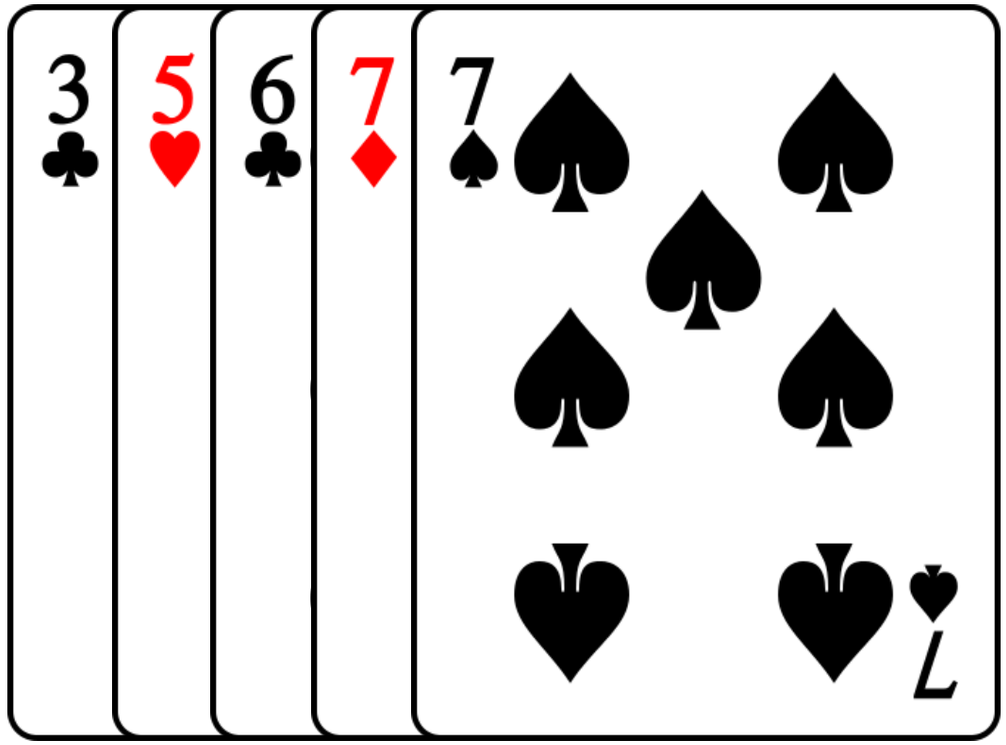
Tri par insertion



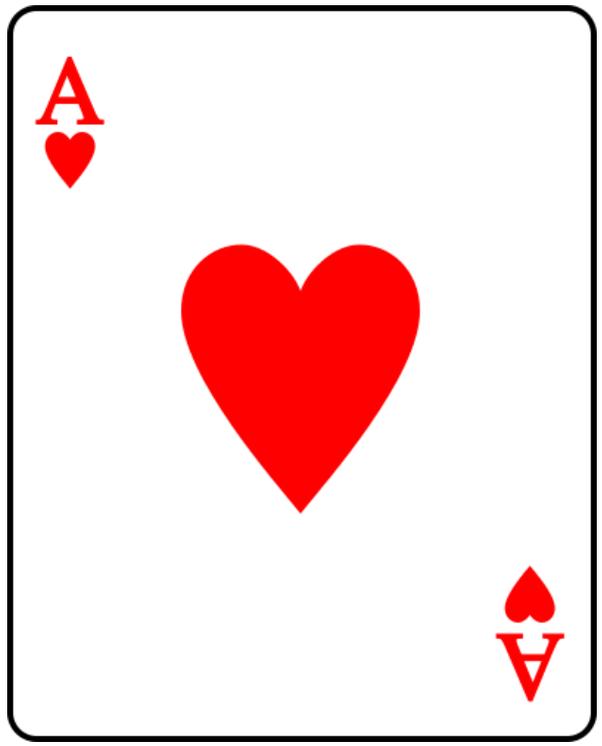
Tri par insertion



Tri par insertion

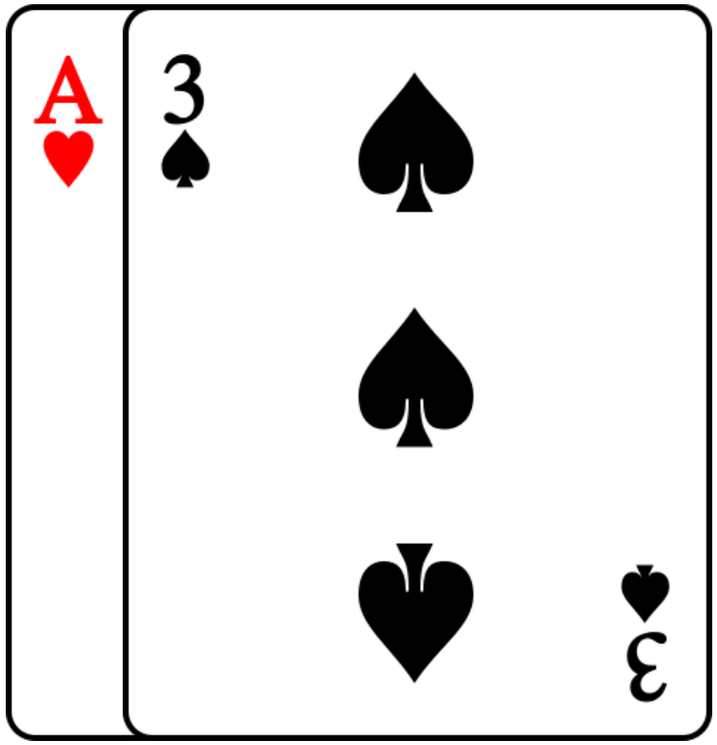


Le meilleur des cas



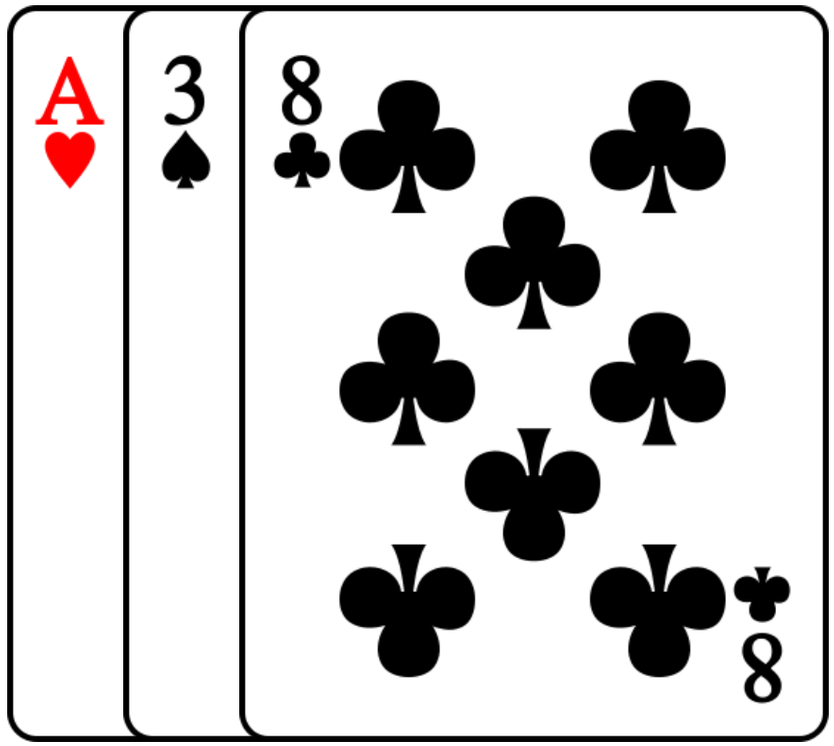
№ operations = 1

Le meilleur des cas



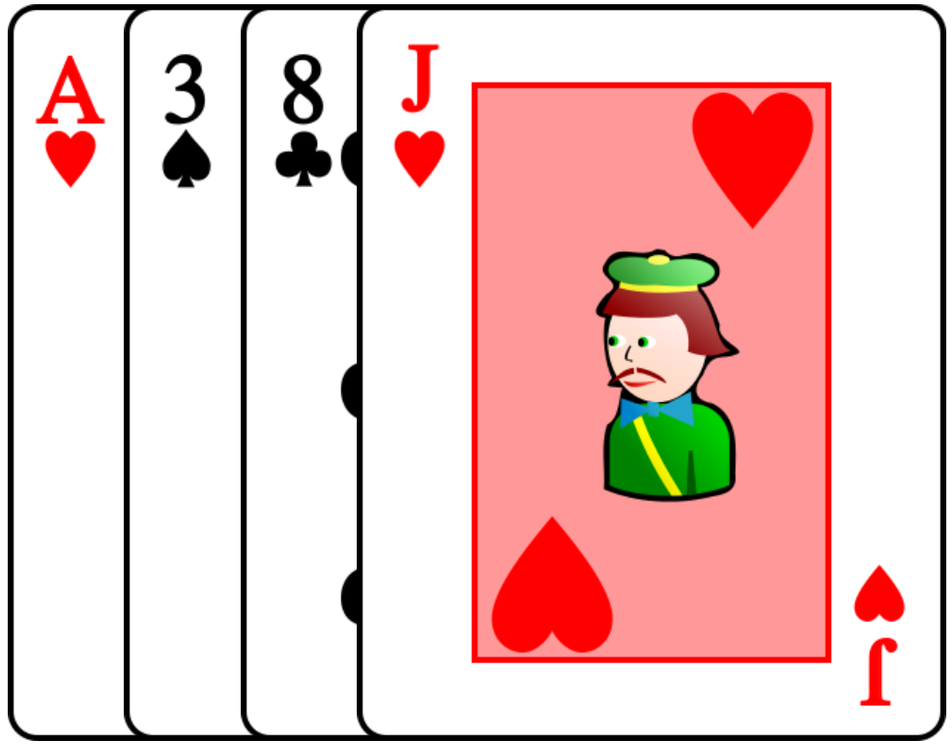
№ operations = 2

Le meilleur des cas



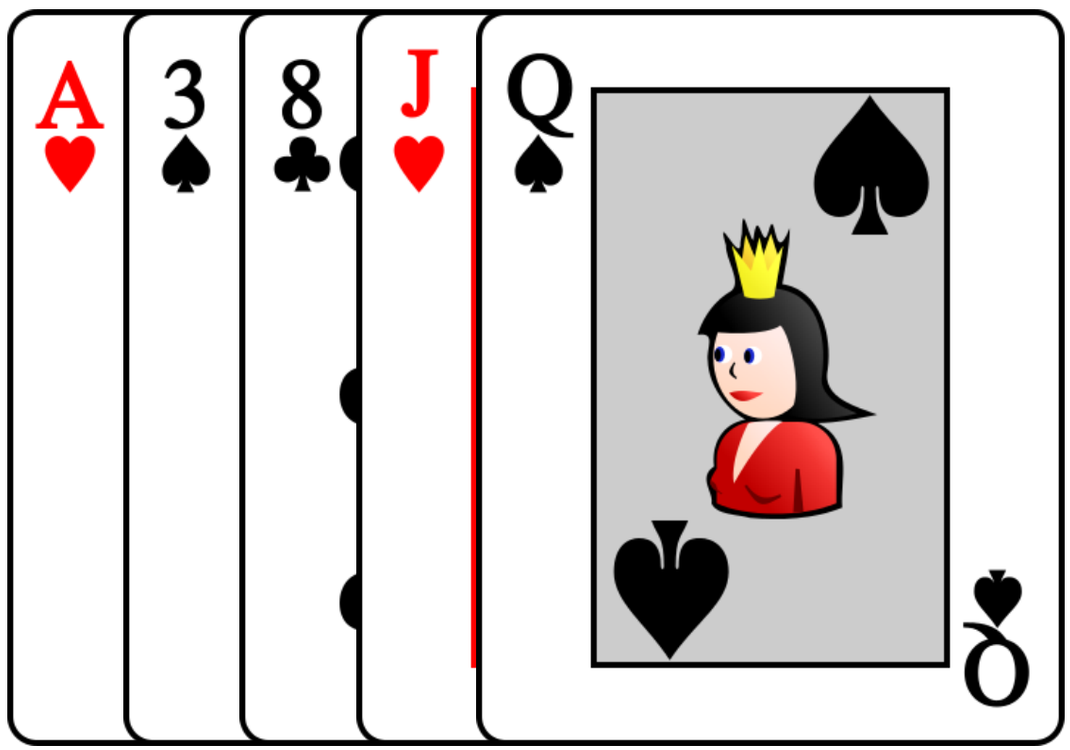
№ operations = 3

Le meilleur des cas



№ operations = 4

Le meilleur des cas

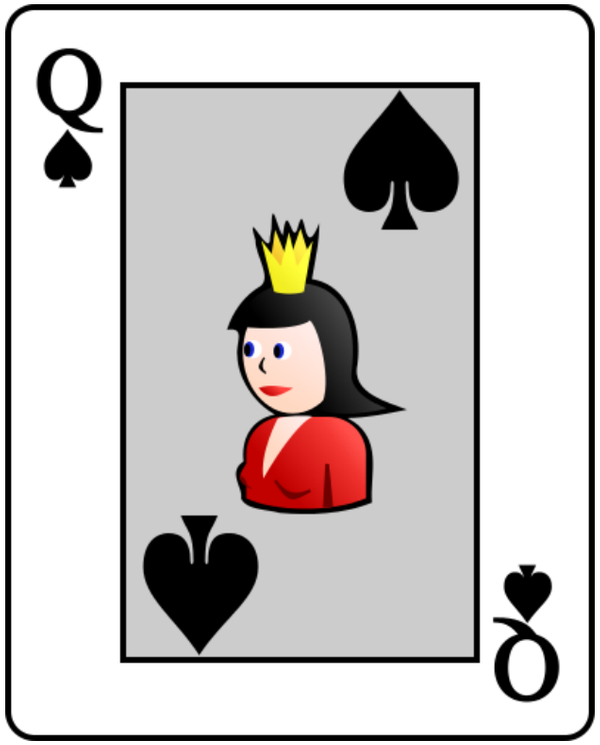


№ operations = 5

Le meilleur des cas

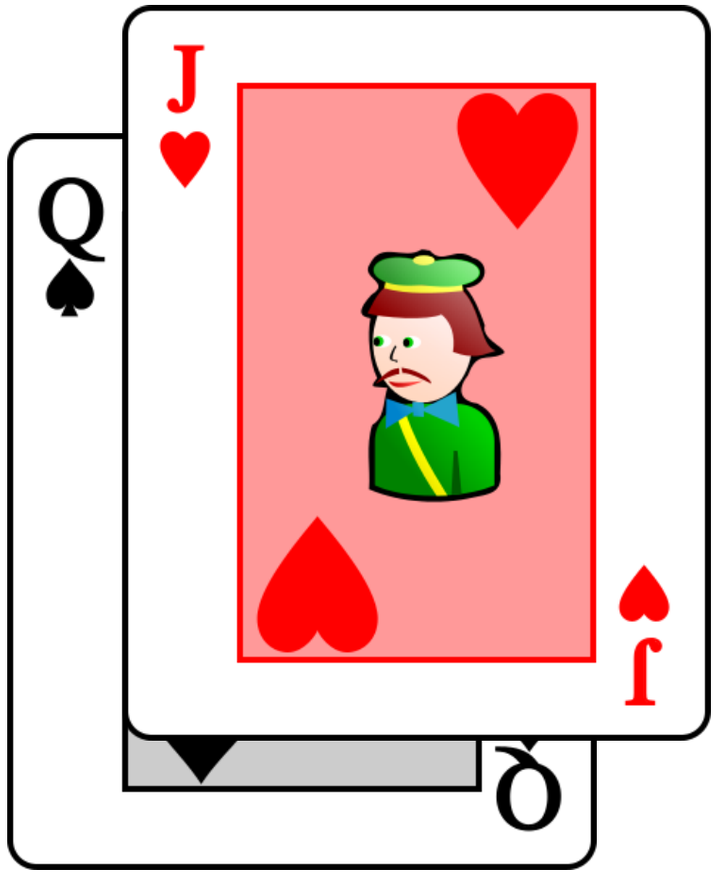
- Les cartes arrivent déjà triées
- On fait n opérations (déplacements de cartes)

Le pire des cas



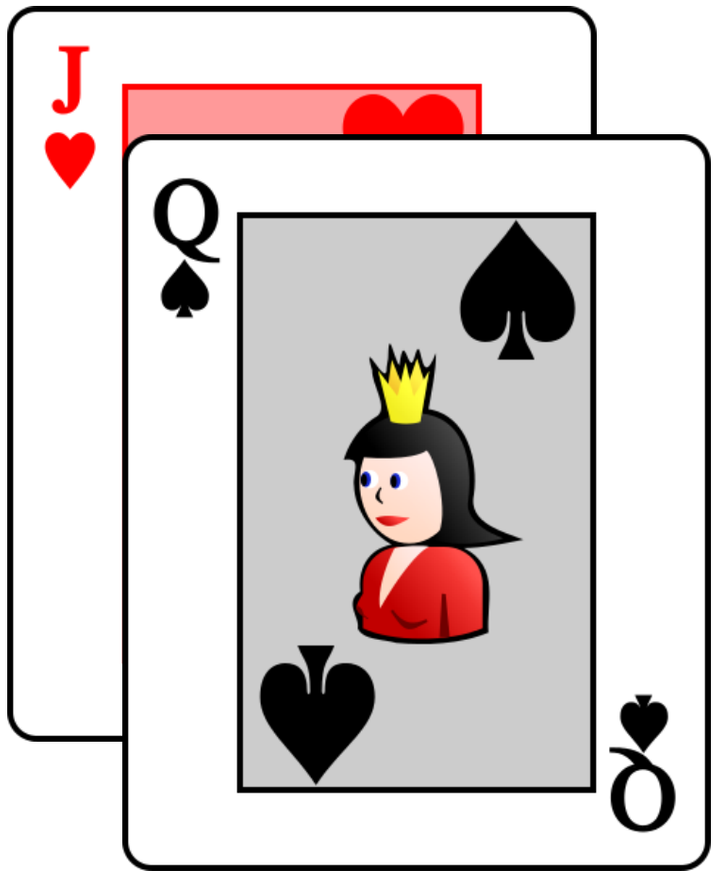
№ operations = 1

Le pire des cas



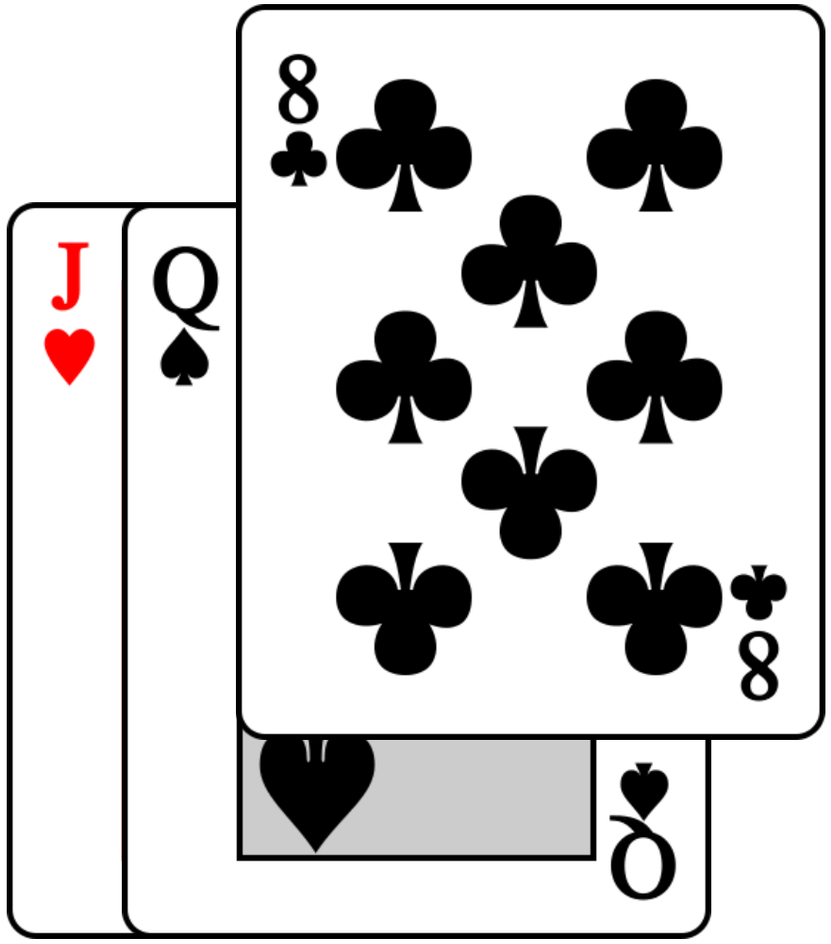
№ operations = 1 + 1

Le pire des cas



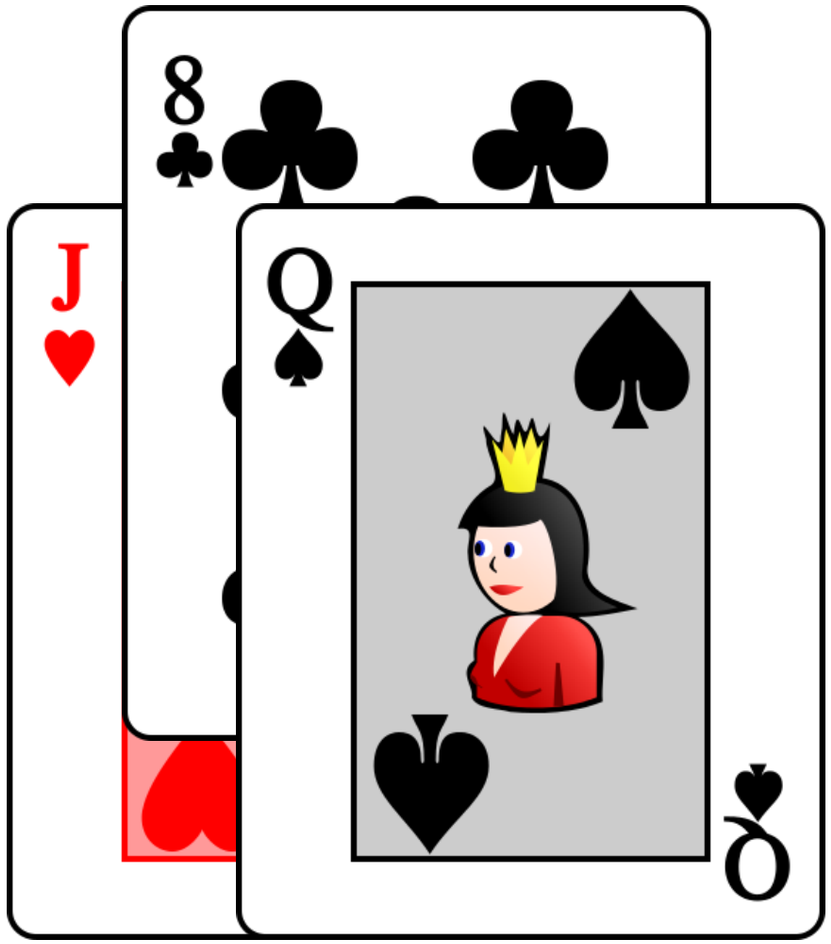
№ operations = 1 + 2

Le pire des cas



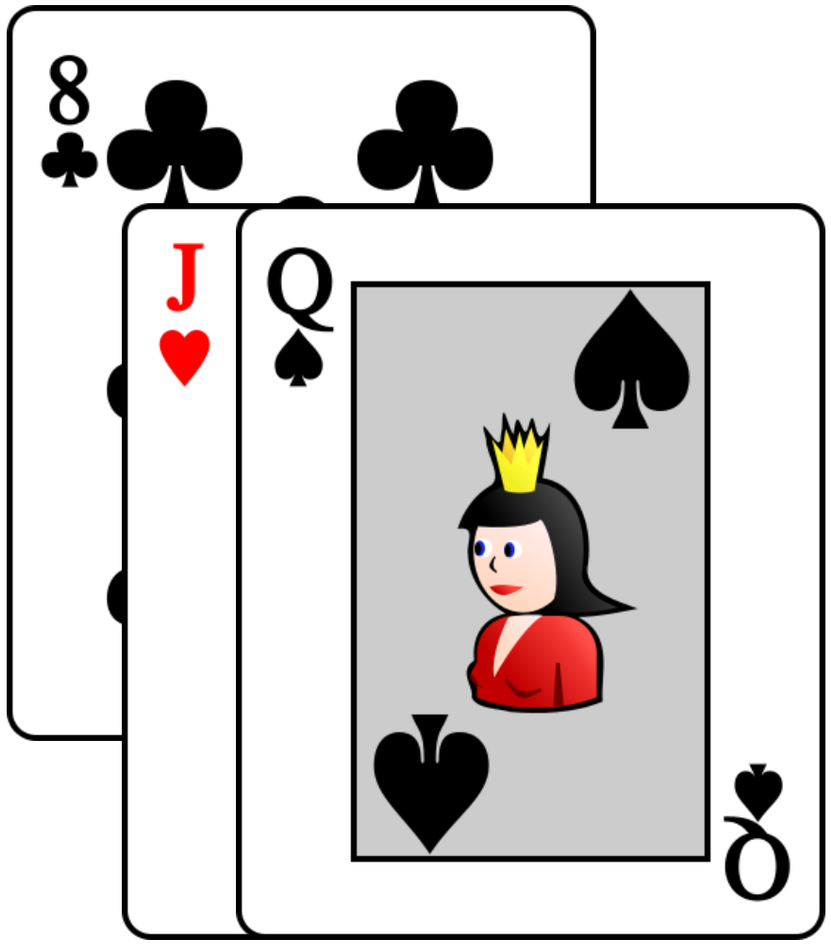
№ operations = 1 + 2 + 1

Le pire des cas



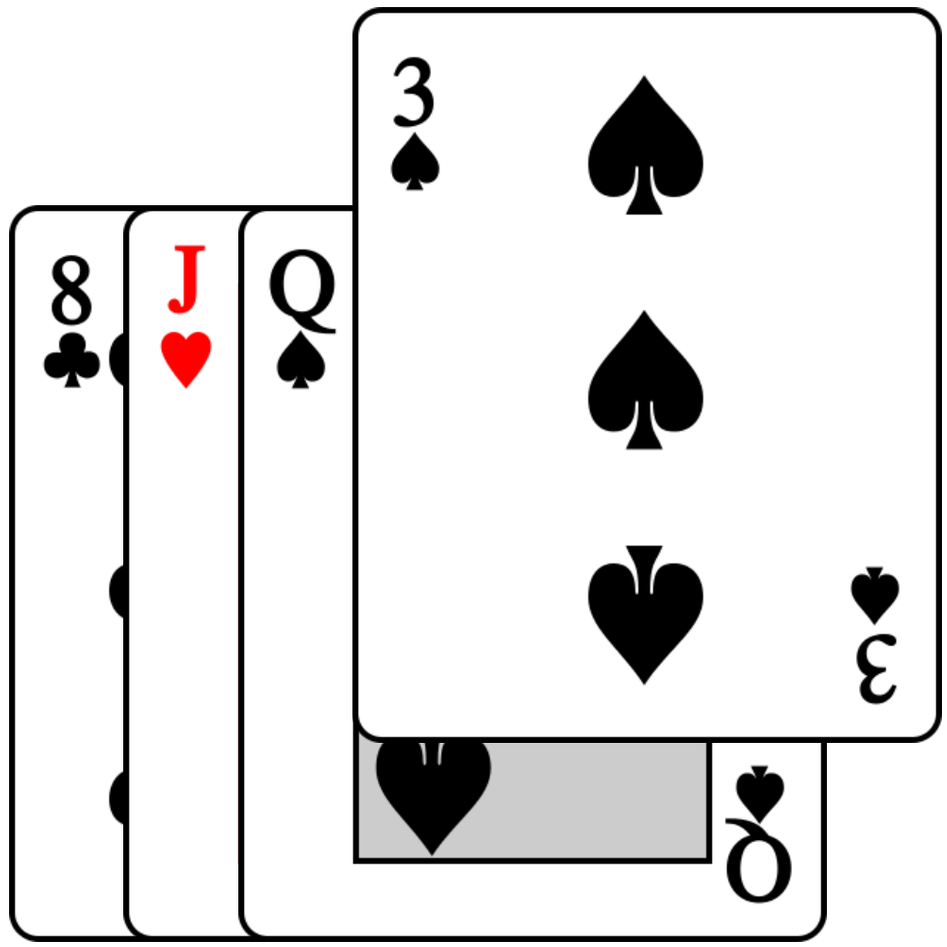
№ operations = 1 + 2 + 2

Le pire des cas



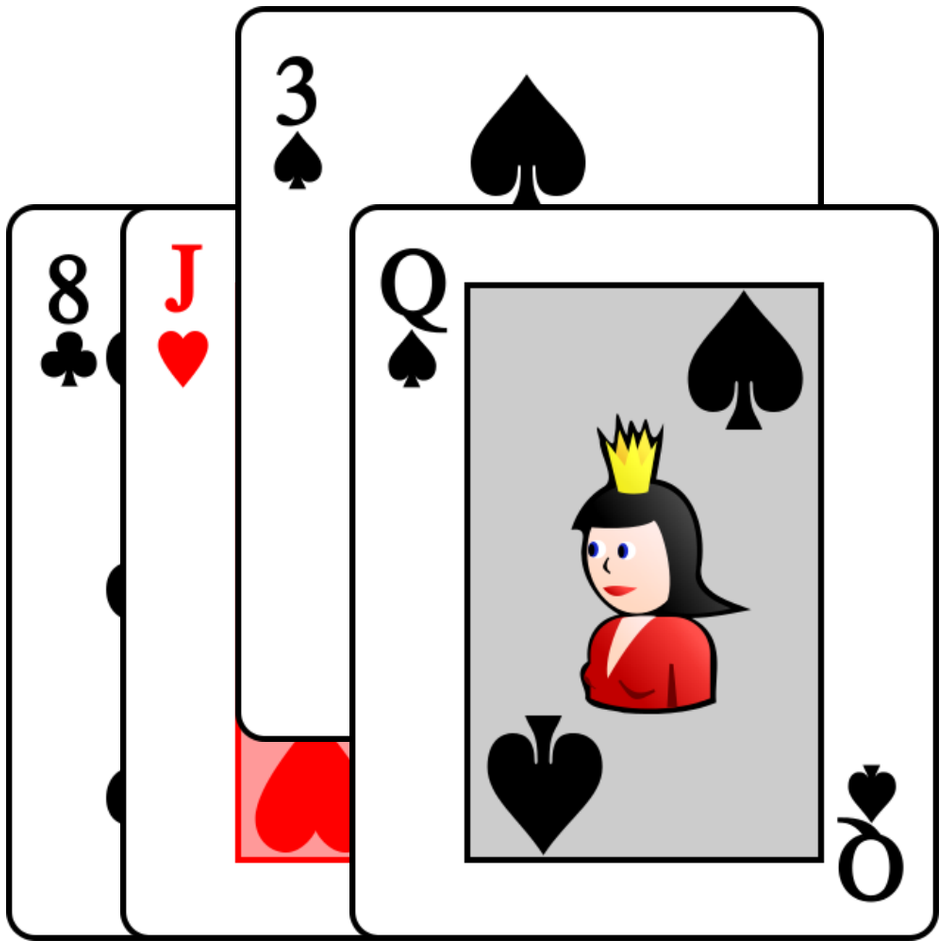
№ operations = 1 + 2 + 3

Le pire des cas



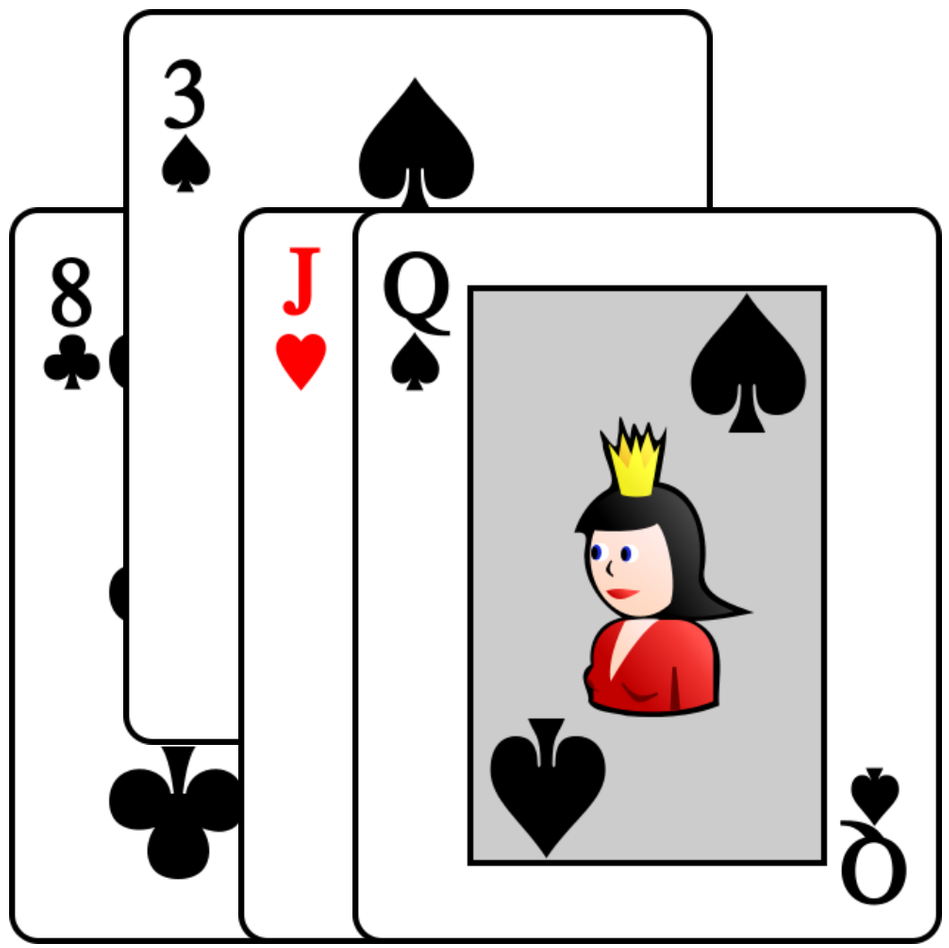
№ operations = 1 + 2 + 3 + 1

Le pire des cas



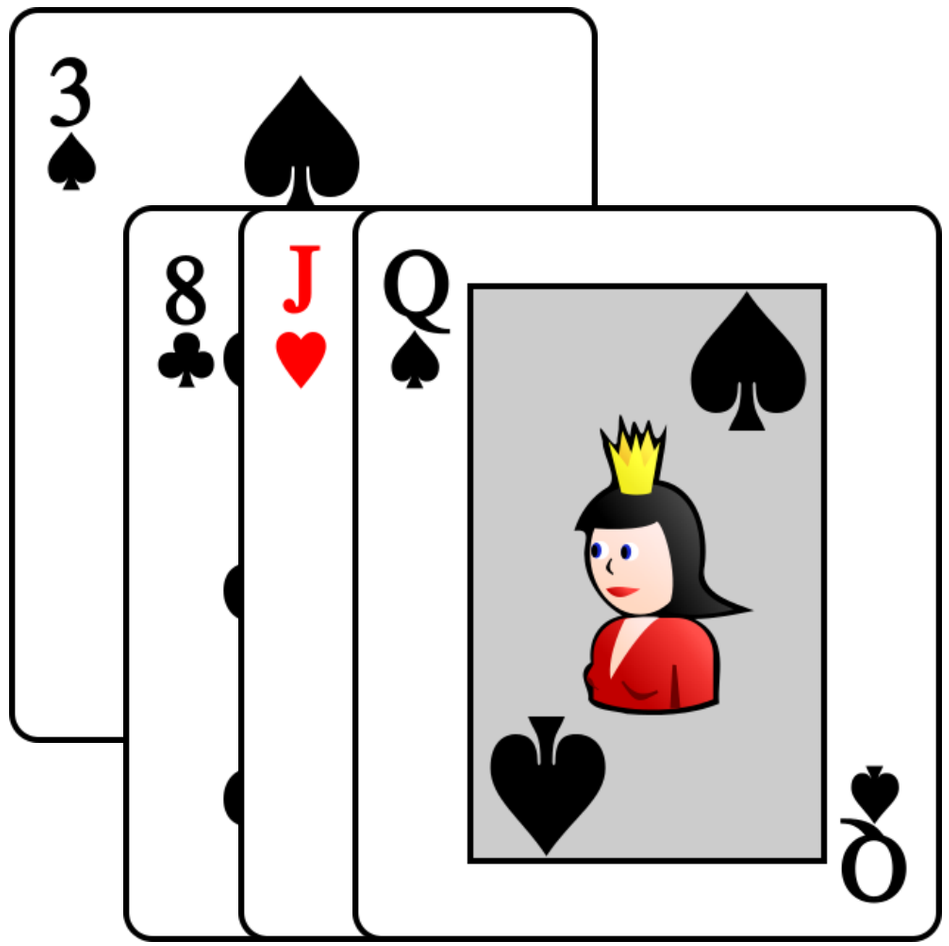
№ operations = 1 + 2 + 3 + 2

Le pire des cas



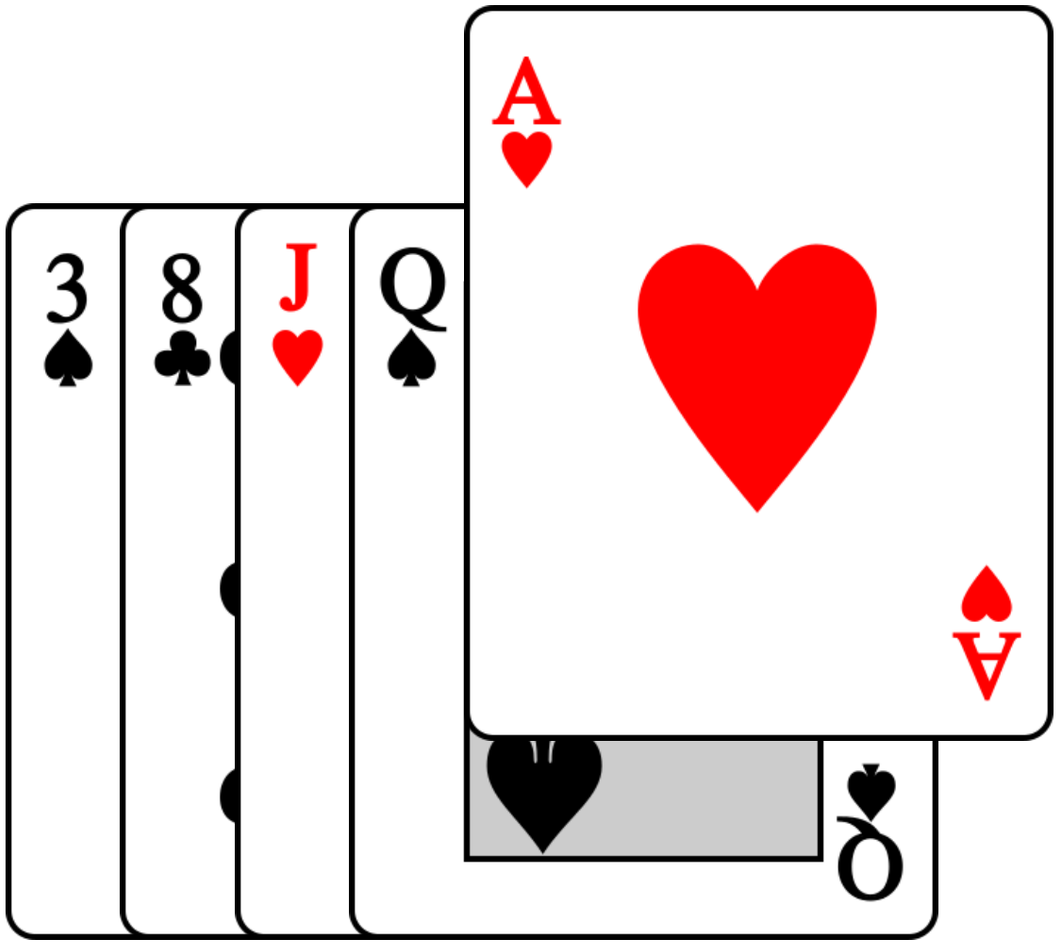
№ operations = 1 + 2 + 3 + 3

Le pire des cas



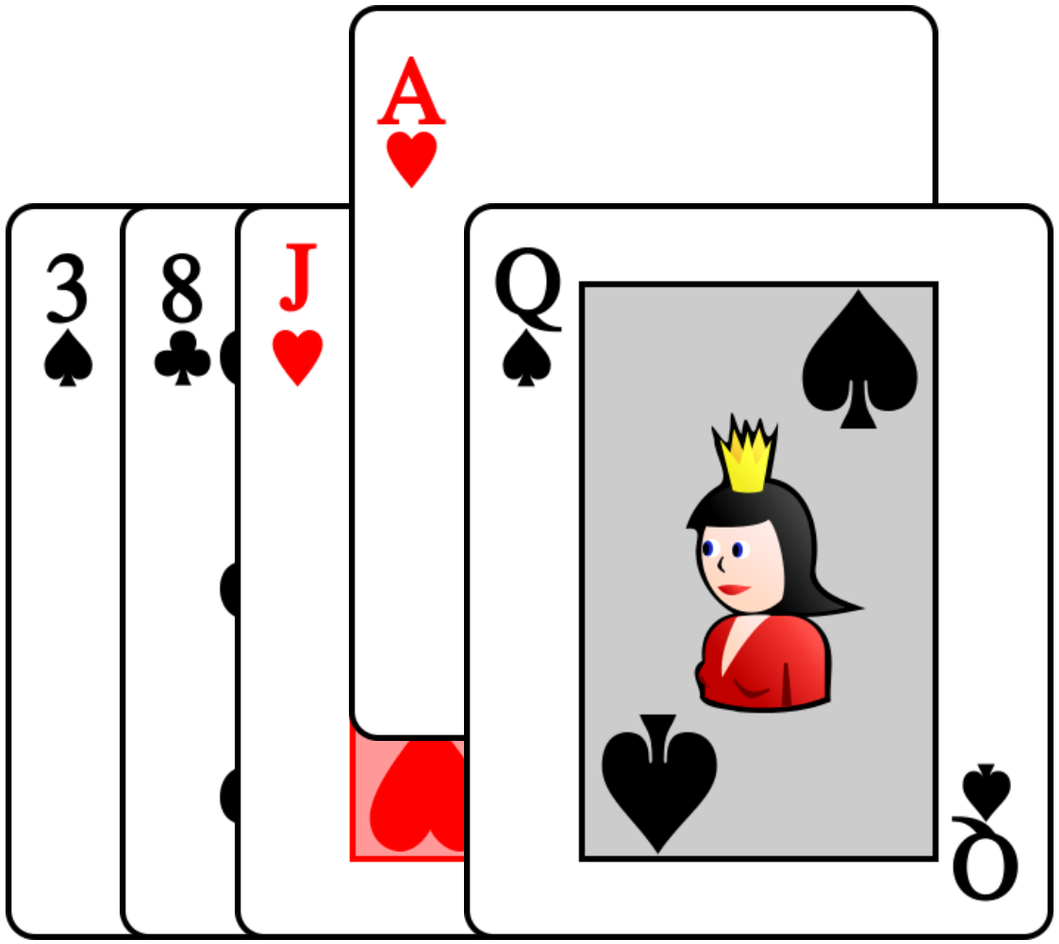
№ operations = 1 + 2 + 3 + 4

Le pire des cas



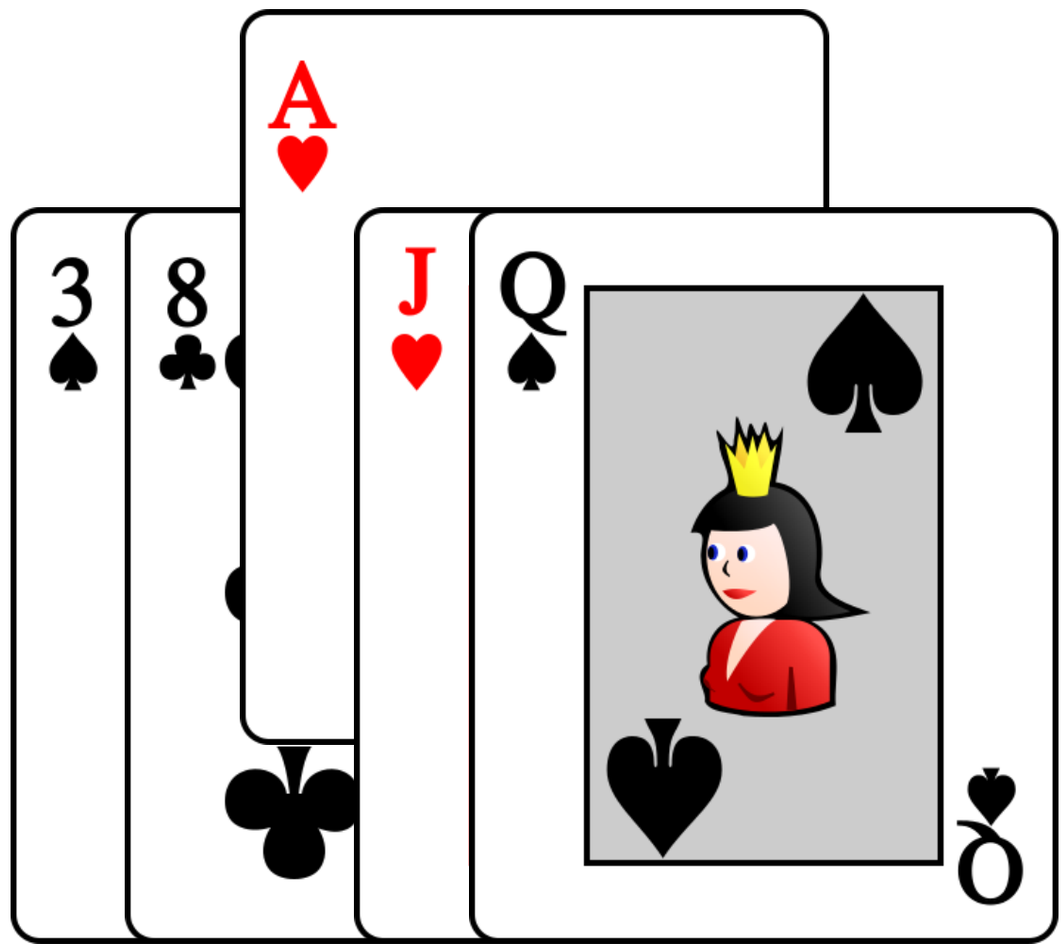
№ operations = 1 + 2 + 3 + 4 + 1

Le pire des cas



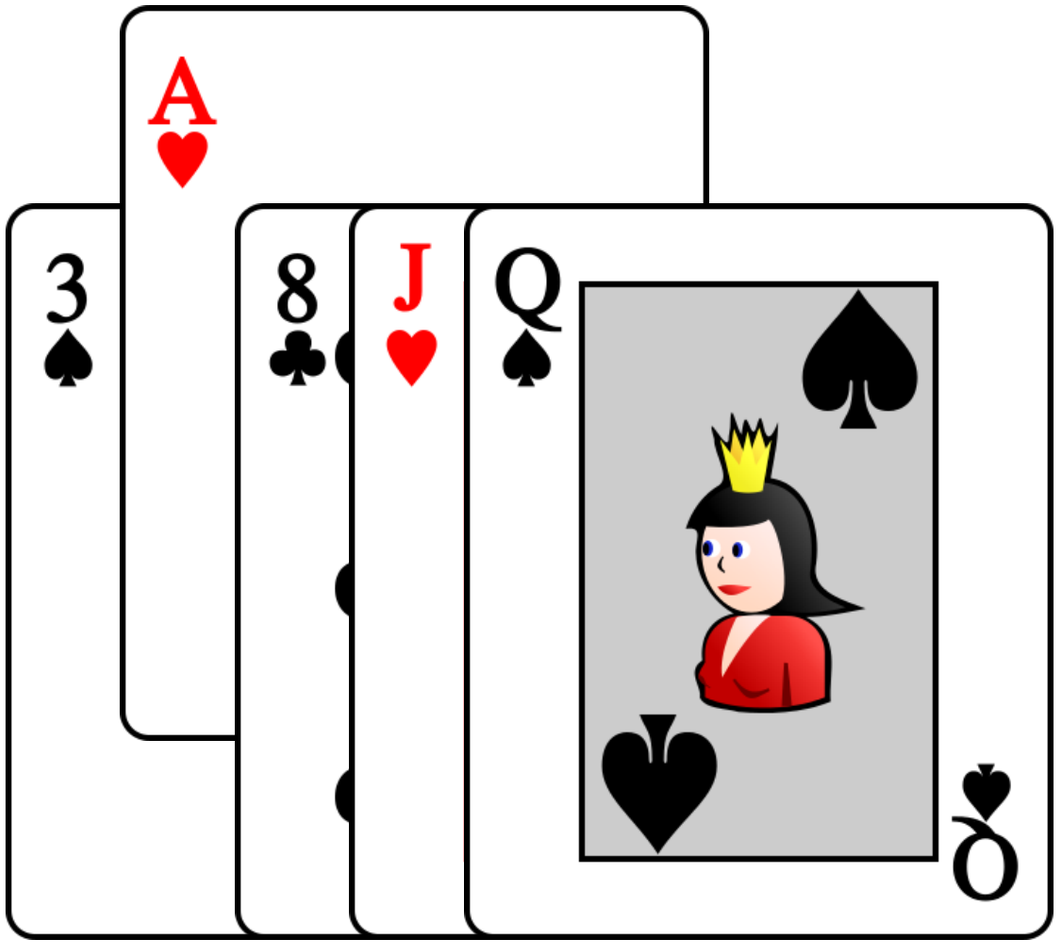
№ operations = 1 + 2 + 3 + 4 + 2

Le pire des cas



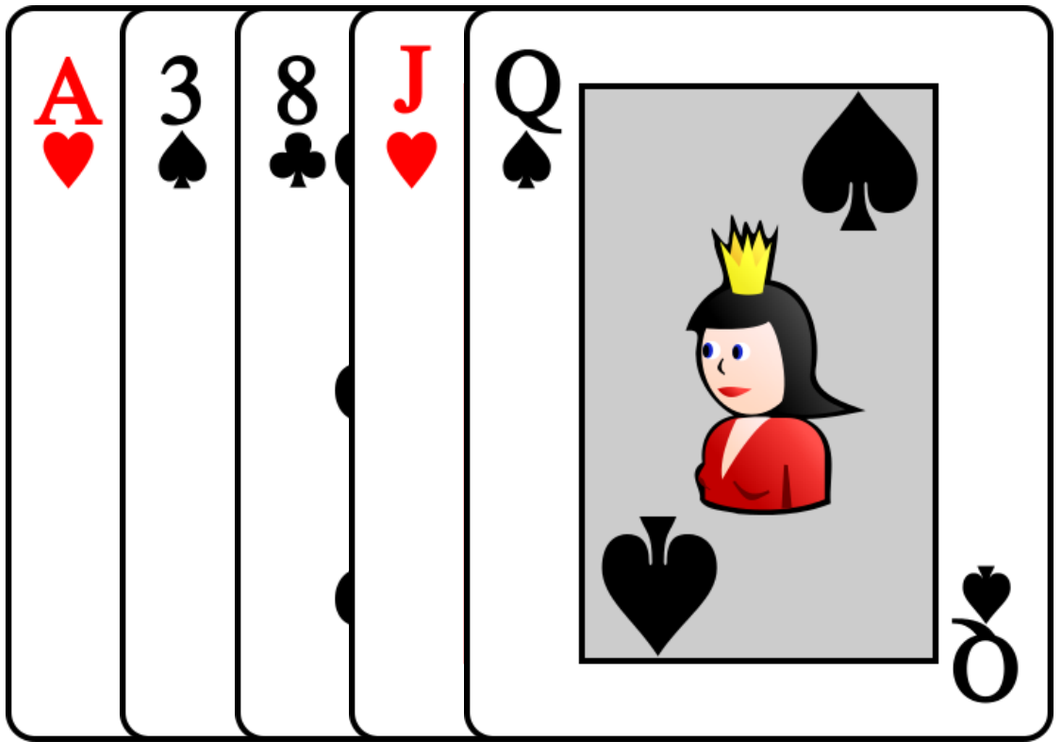
№ operations = 1 + 2 + 3 + 4 + 3

Le pire des cas



№ operations = 1 + 2 + 3 + 4 + 4

Le pire des cas



№ operations = 1 + 2 + 3 + 4 + 5

Le pire des cas

- Les cartes arrivent en ordre décroissant
- On fait i opérations pour la i-ème carte
- Le nombre totale est 1 + 2 + 3 + ⋯ + n

$$\sum_{i=1}^{n} i = \frac{1}{2}n(n+1) = \frac{1}{2}(n^2+n) = \frac{1}{2}n^2 + \frac{1}{2}n \in O(n^2)$$

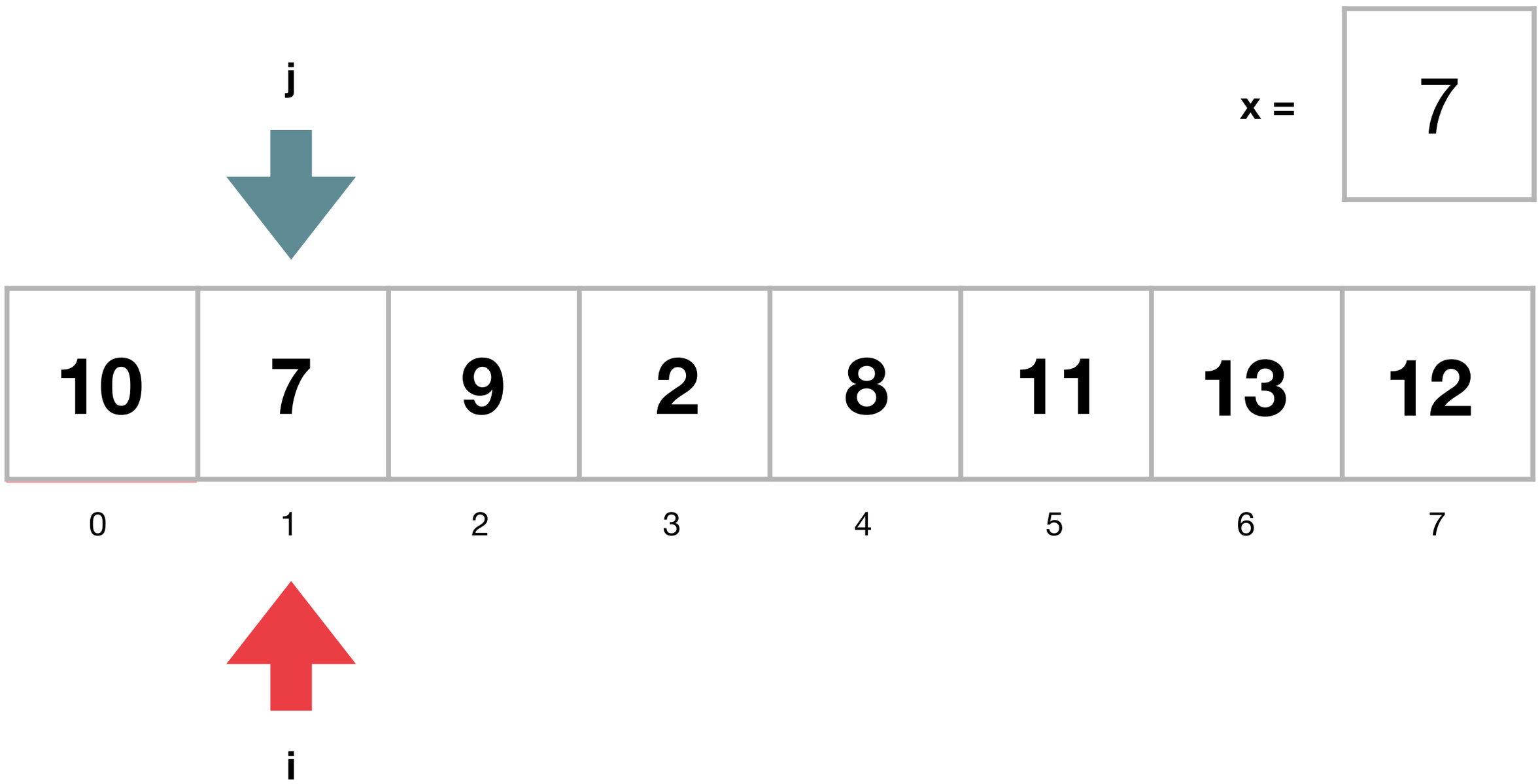
Tri d’un tableau par insertion

```
procedure trier-par-insertion(T)
    n ≔ longueur(T)
    pour i ≔ 1 à n – 1 faire (on saute le 0)
        x ≔ T[i]
        j ≔ i
        tant que j > 0 et x < T[j – 1] faire
            (décaler d’un élément)
            T[j] ≔ T[j – 1]
            j ≔ j – 1
        fin tant que
        (ici x ≥ T[j – 1] ou bien j = 0)
        T[j] ≔ x
    fin pour
fin procedure
```

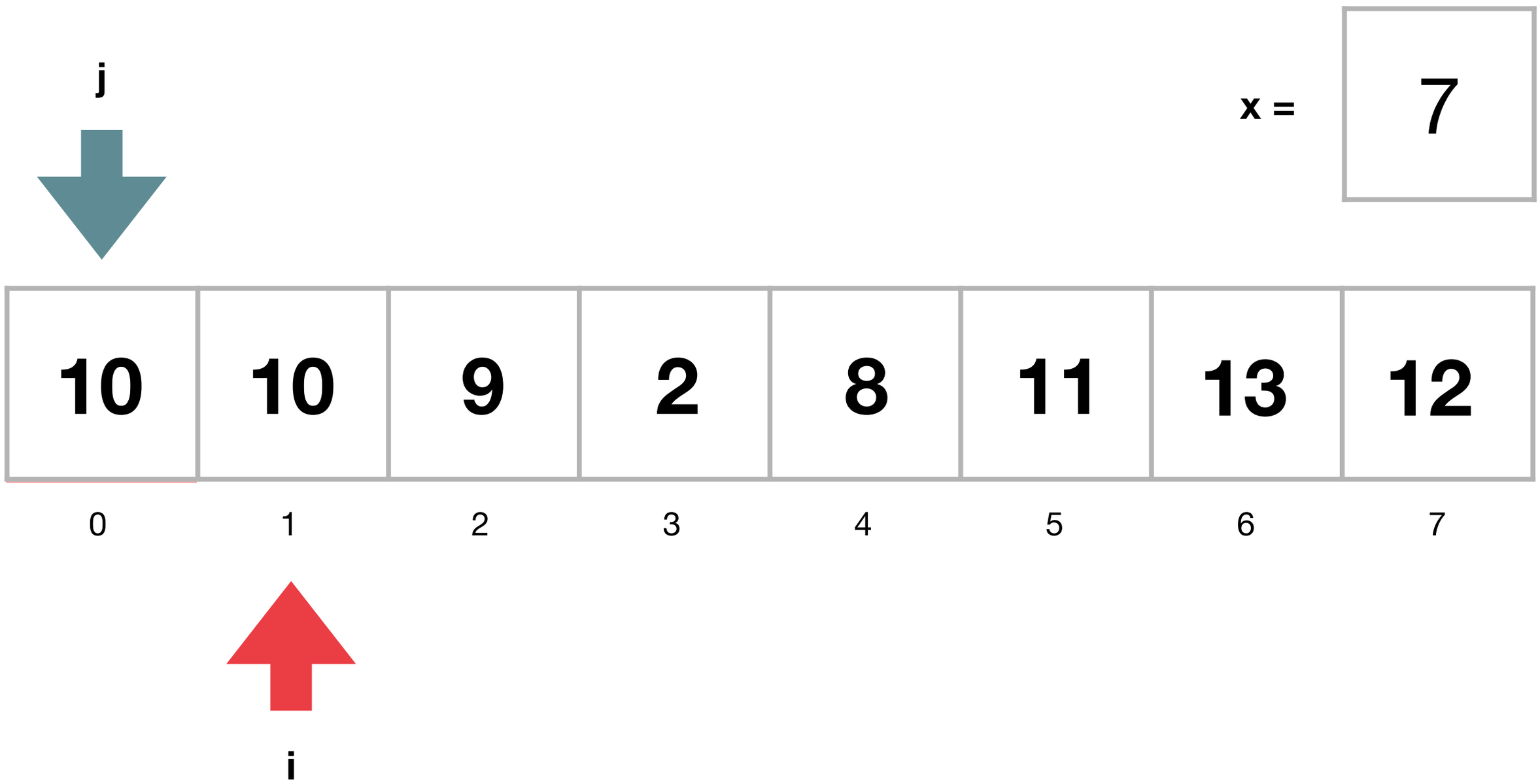
Tri par insertion

10	**7**	**9**	**2**	**8**	**11**	**13**	**12**
0	1	2	3	4	5	6	7

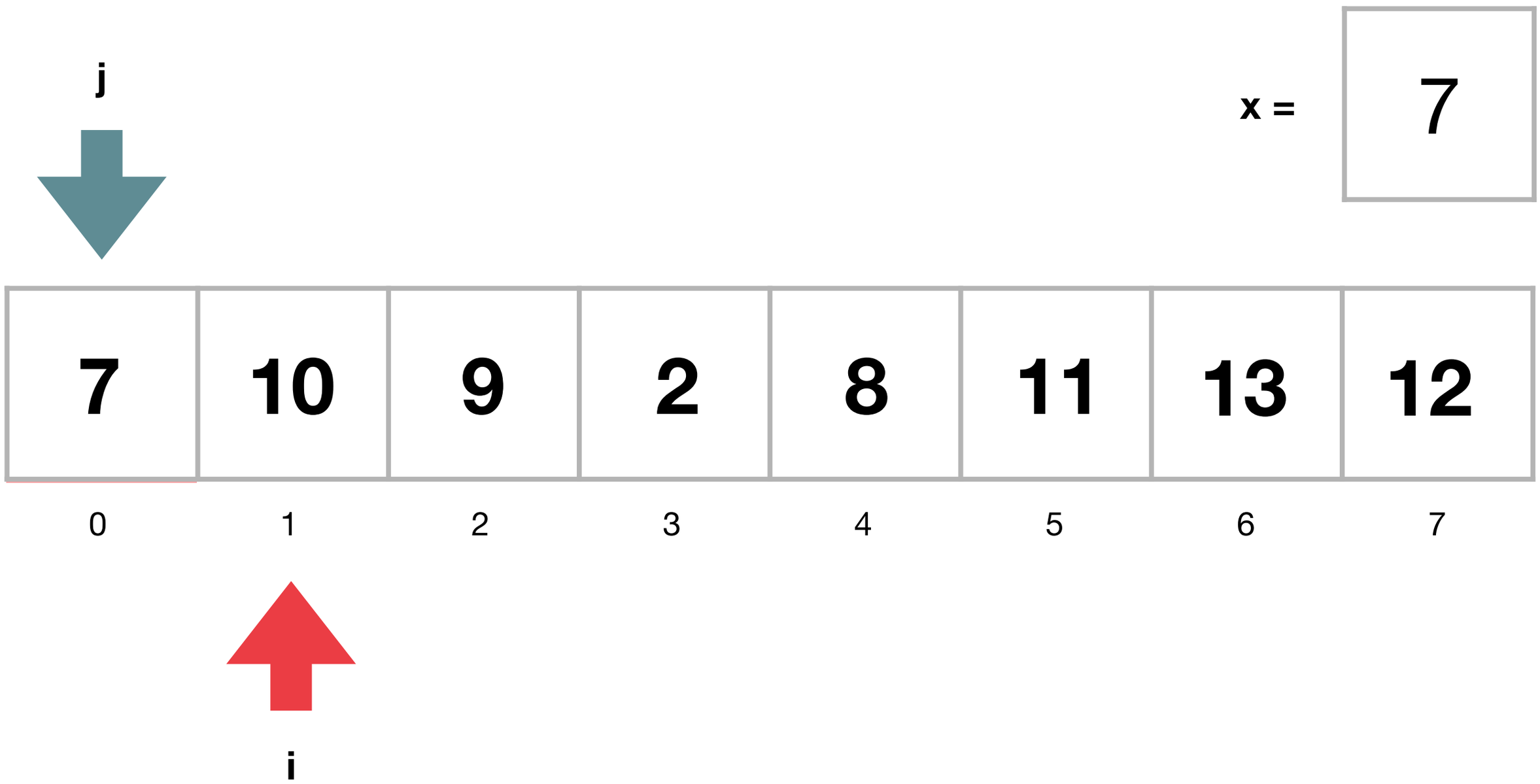
Tri par insertion



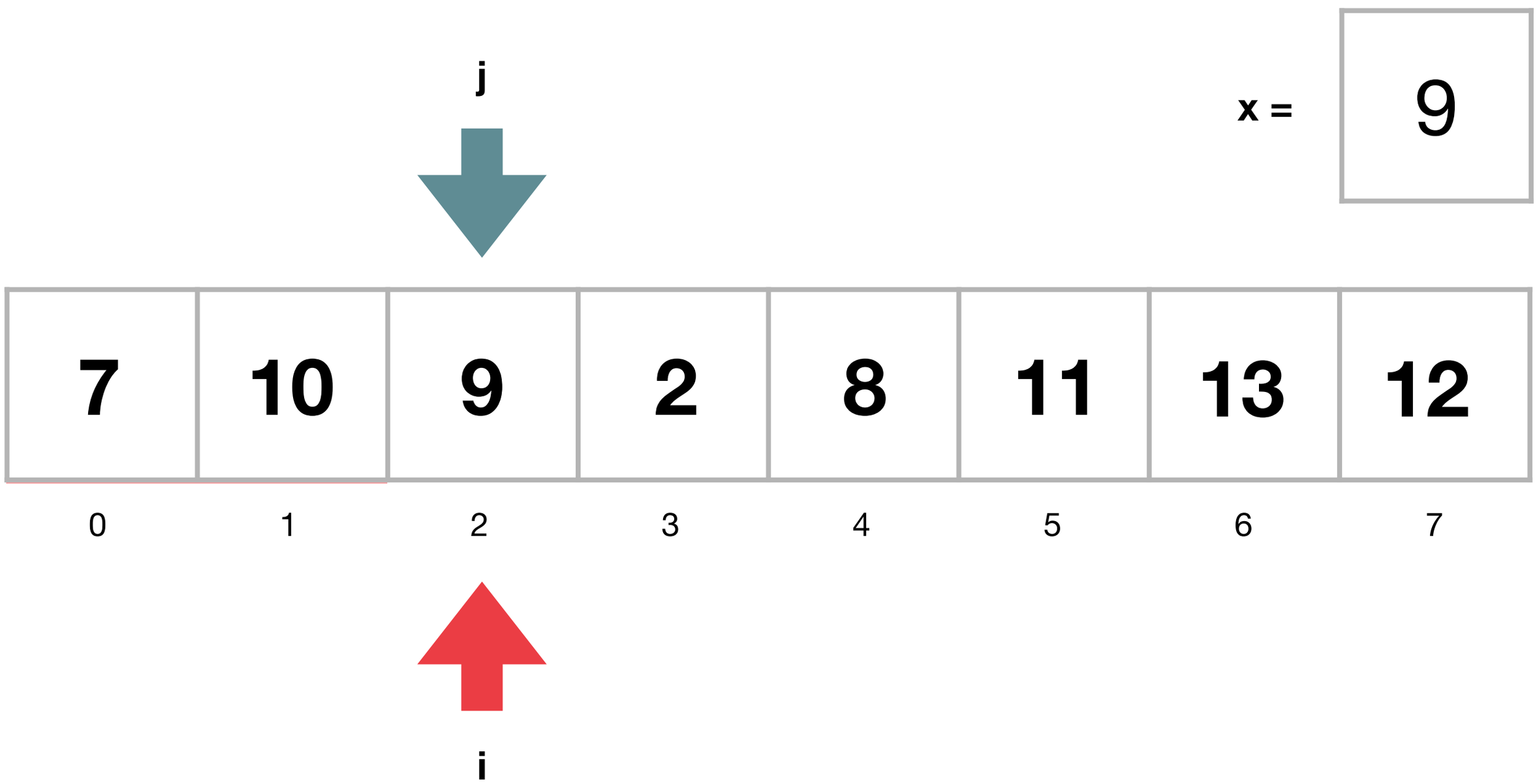
Tri par insertion



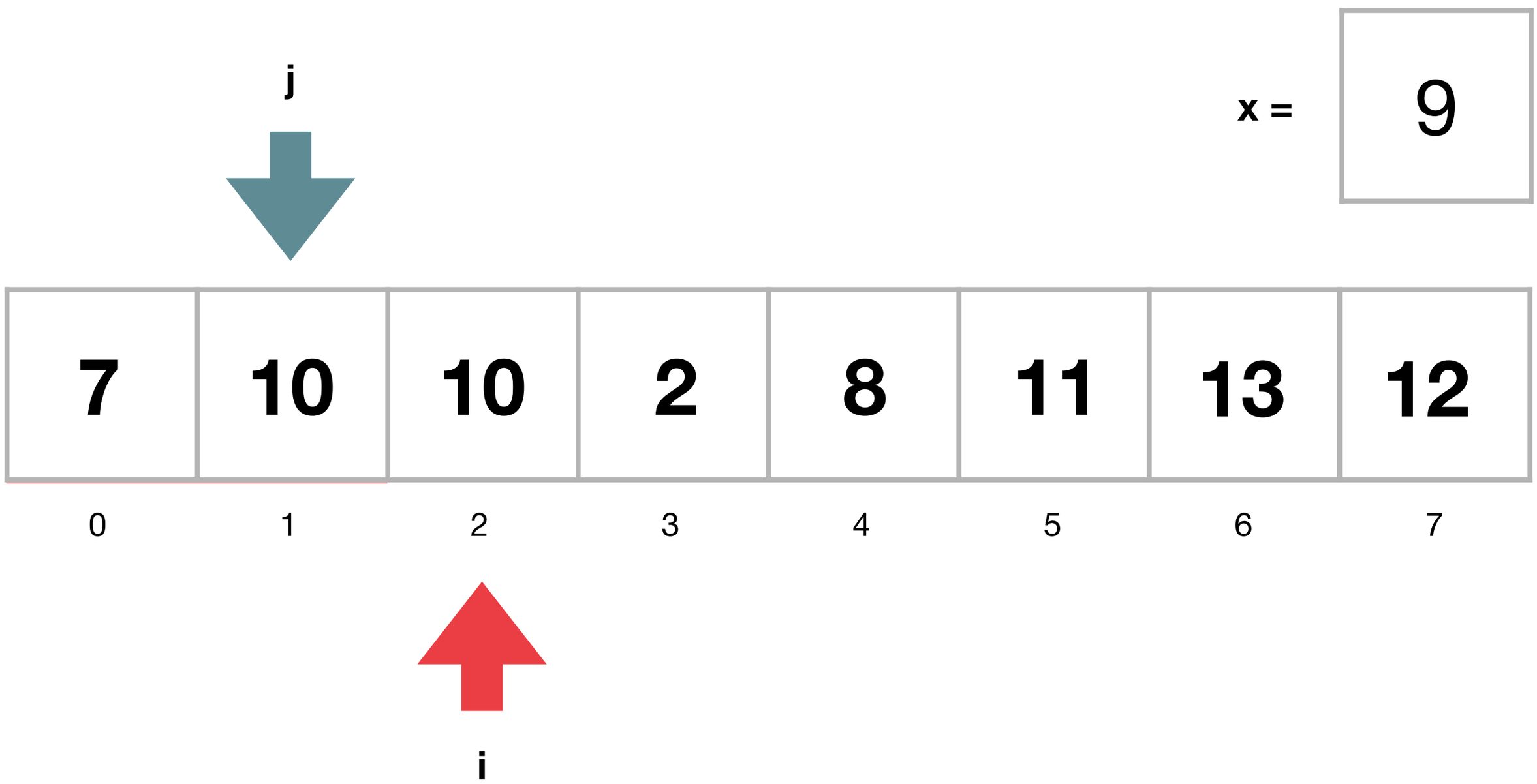
Tri par insertion



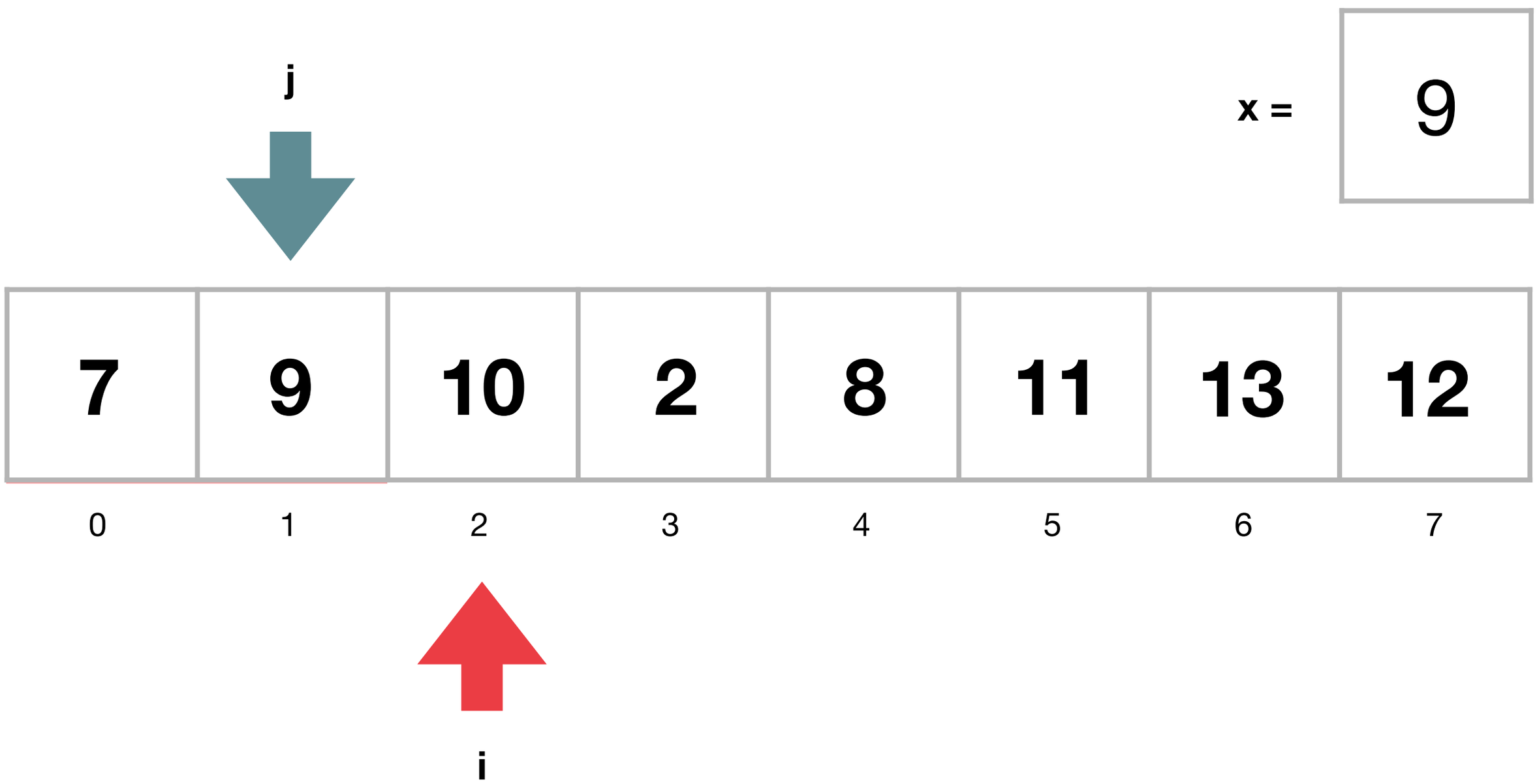
Tri par insertion



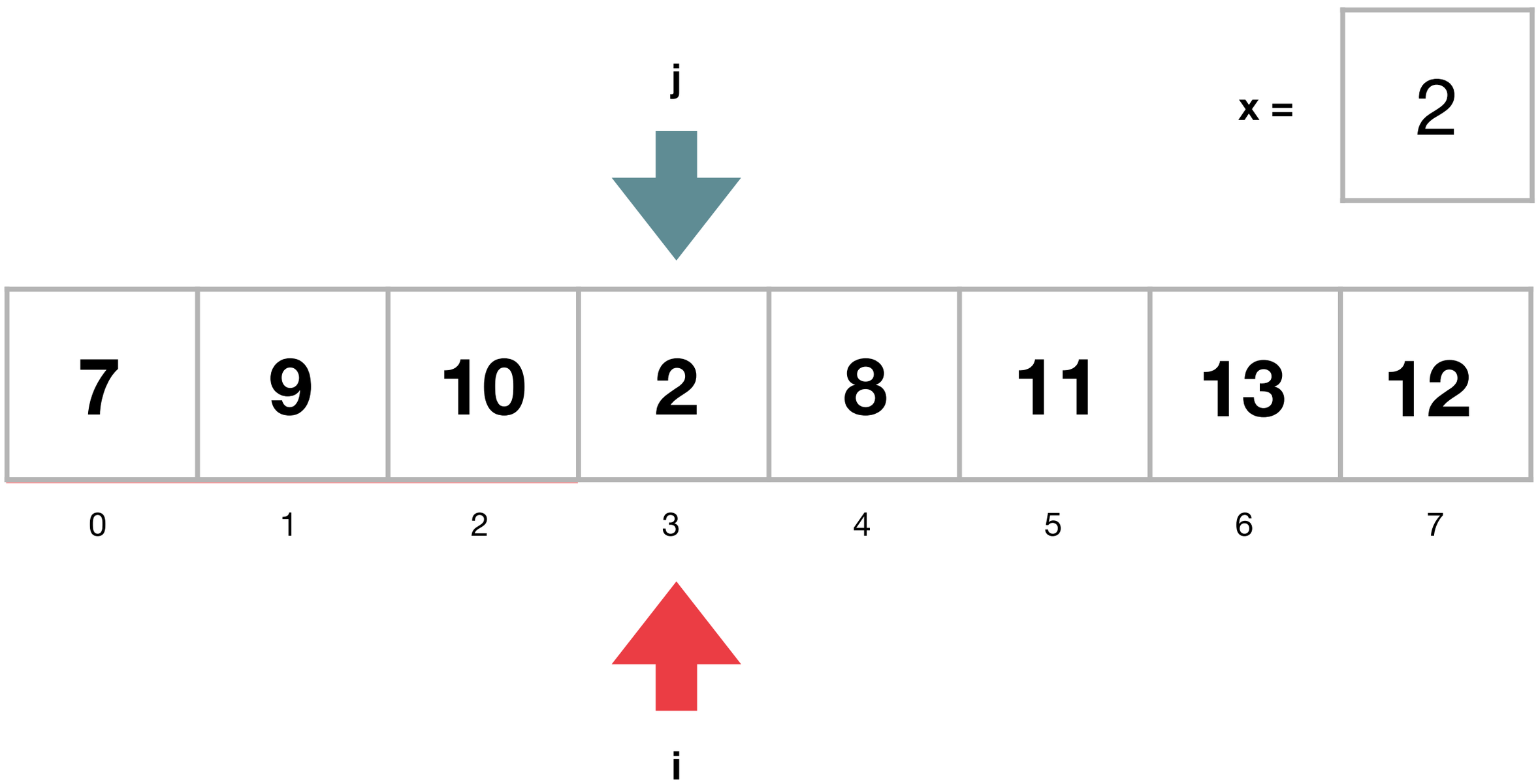
Tri par insertion



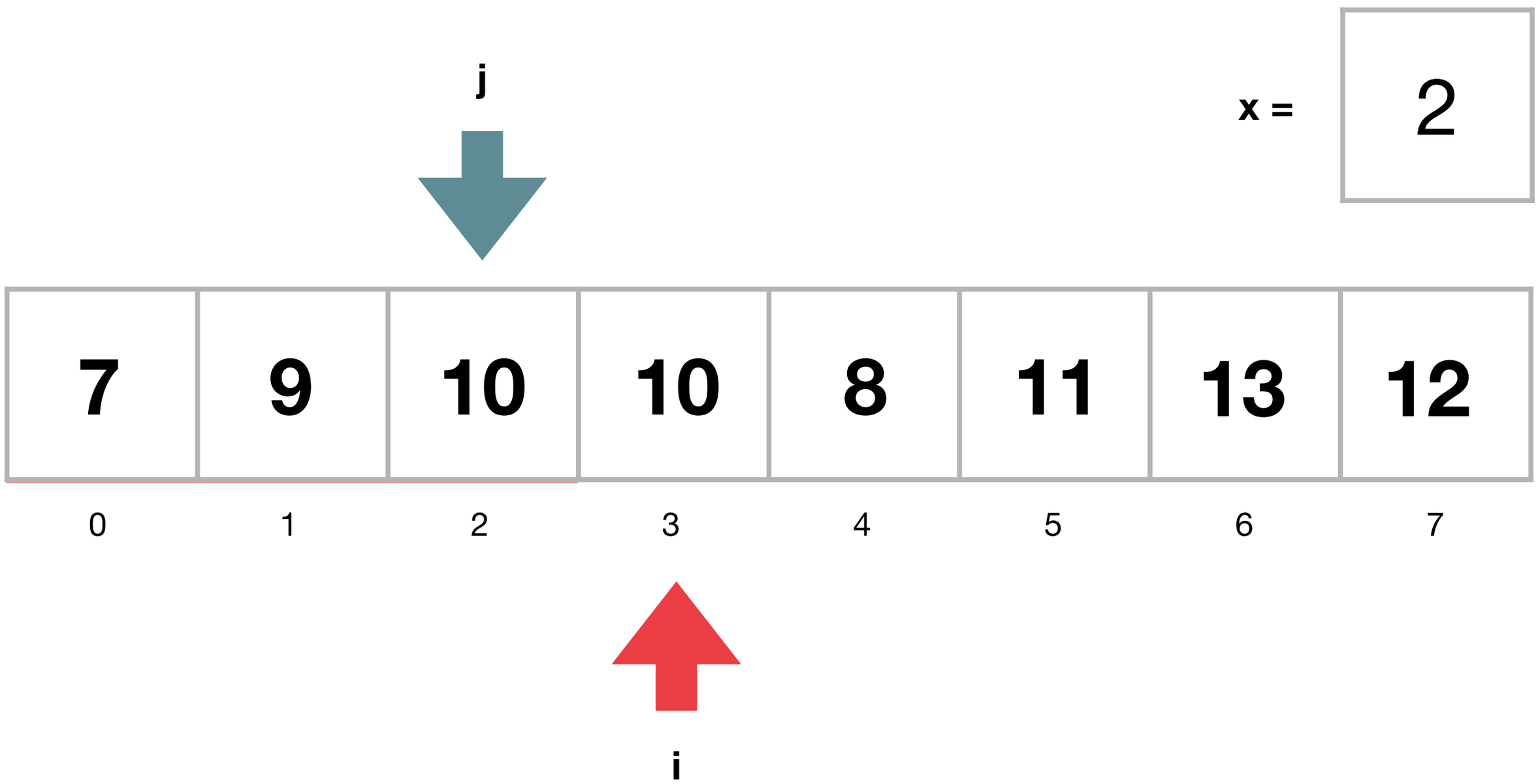
Tri par insertion



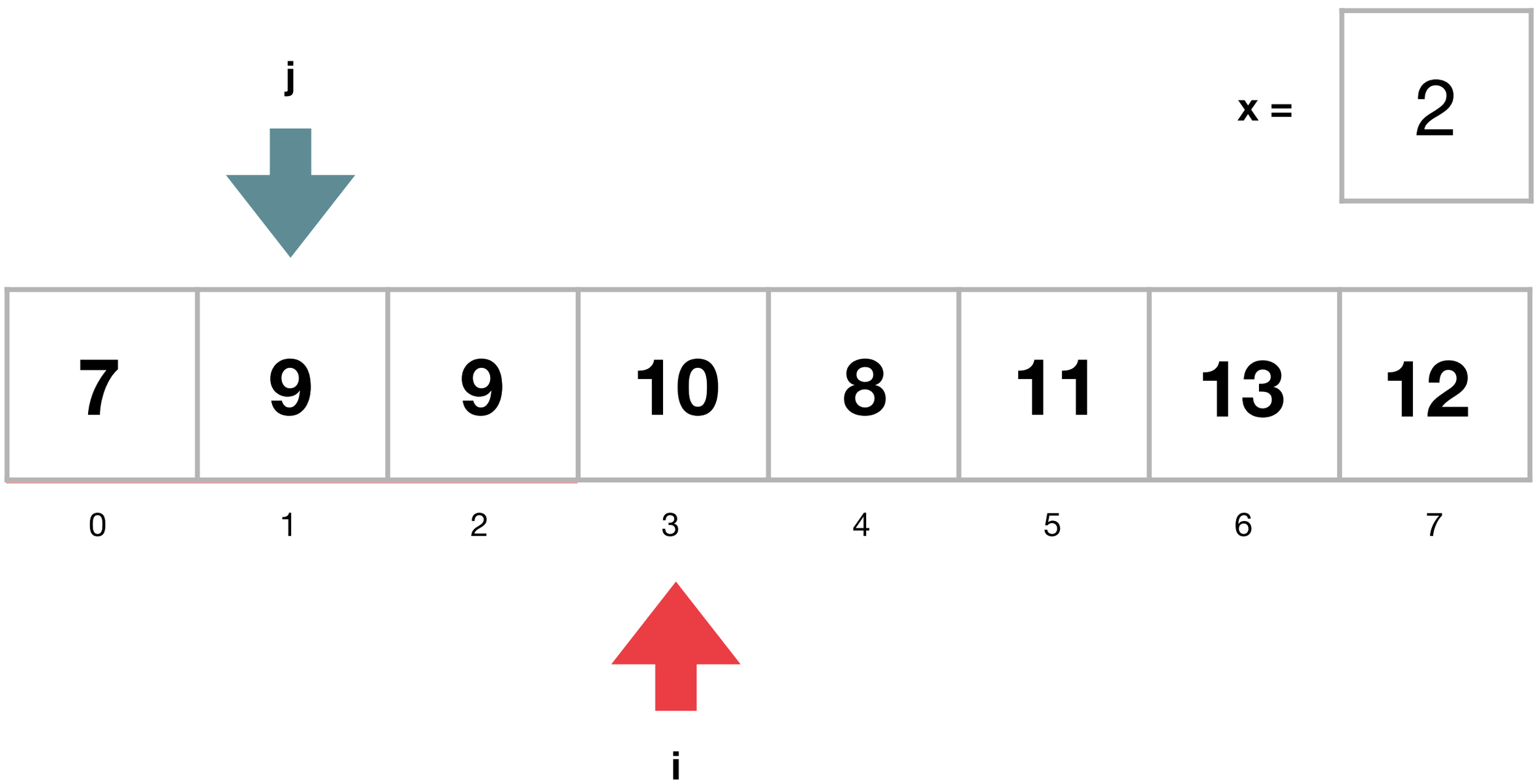
Tri par insertion



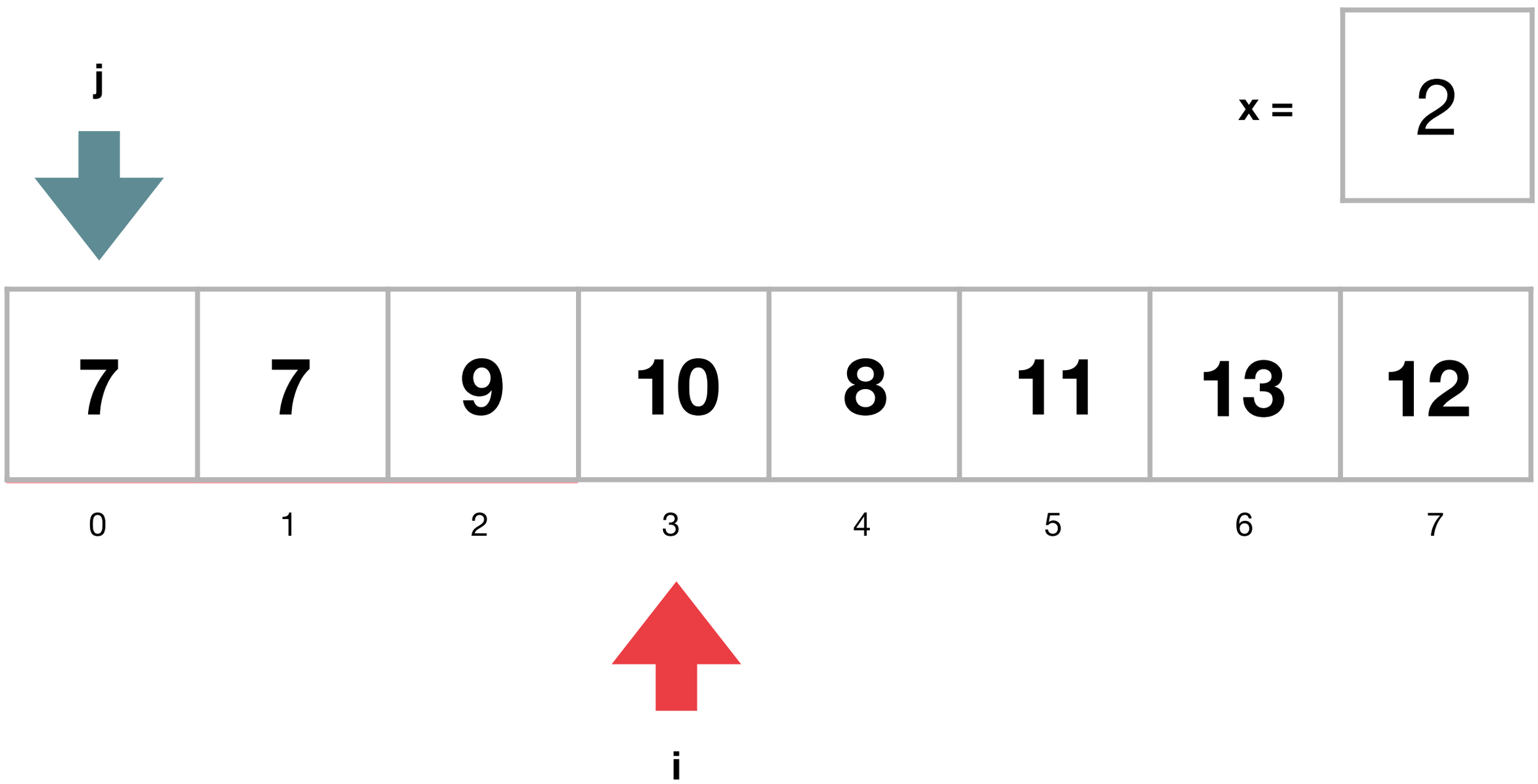
Tri par insertion



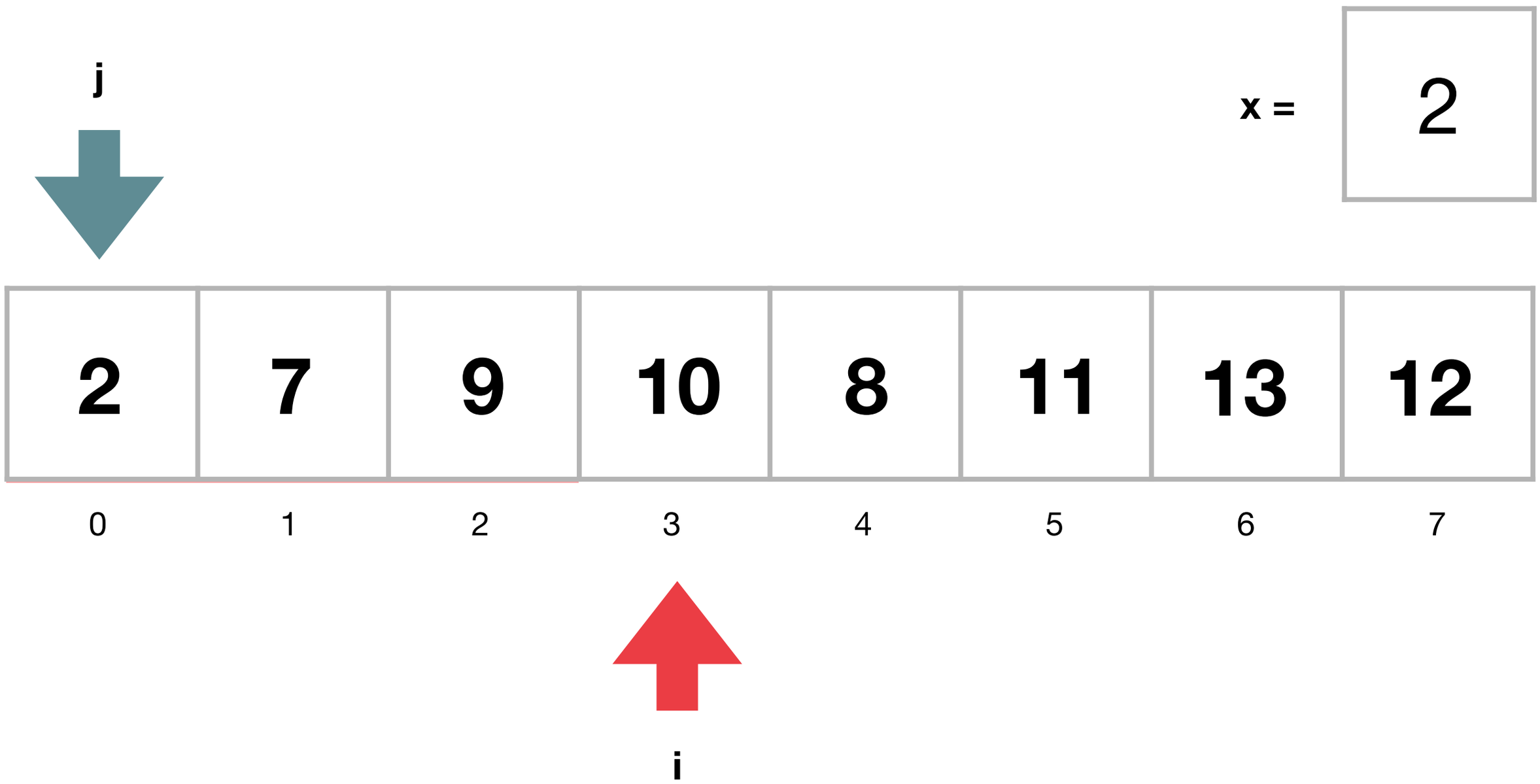
Tri par insertion



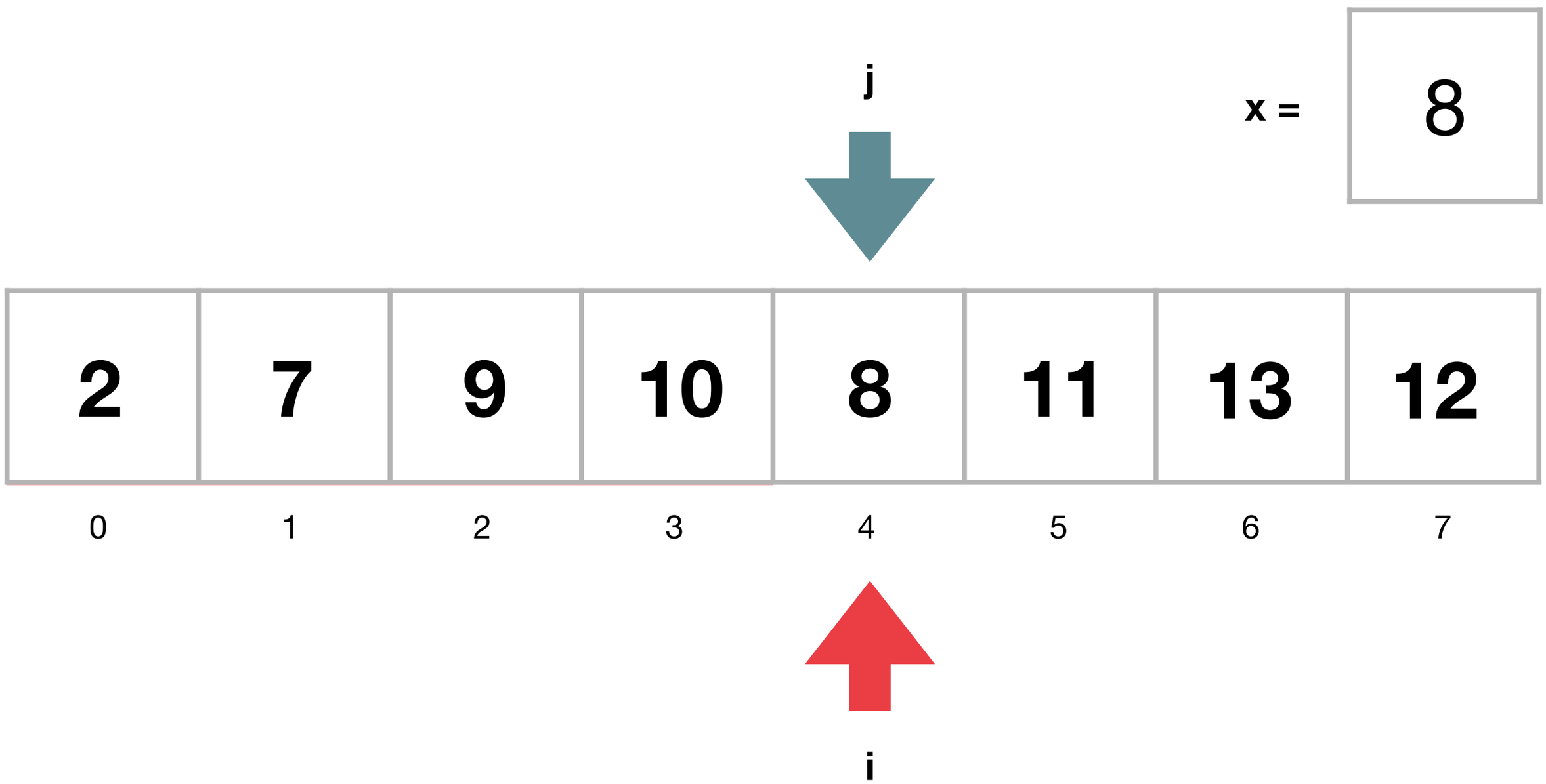
Tri par insertion



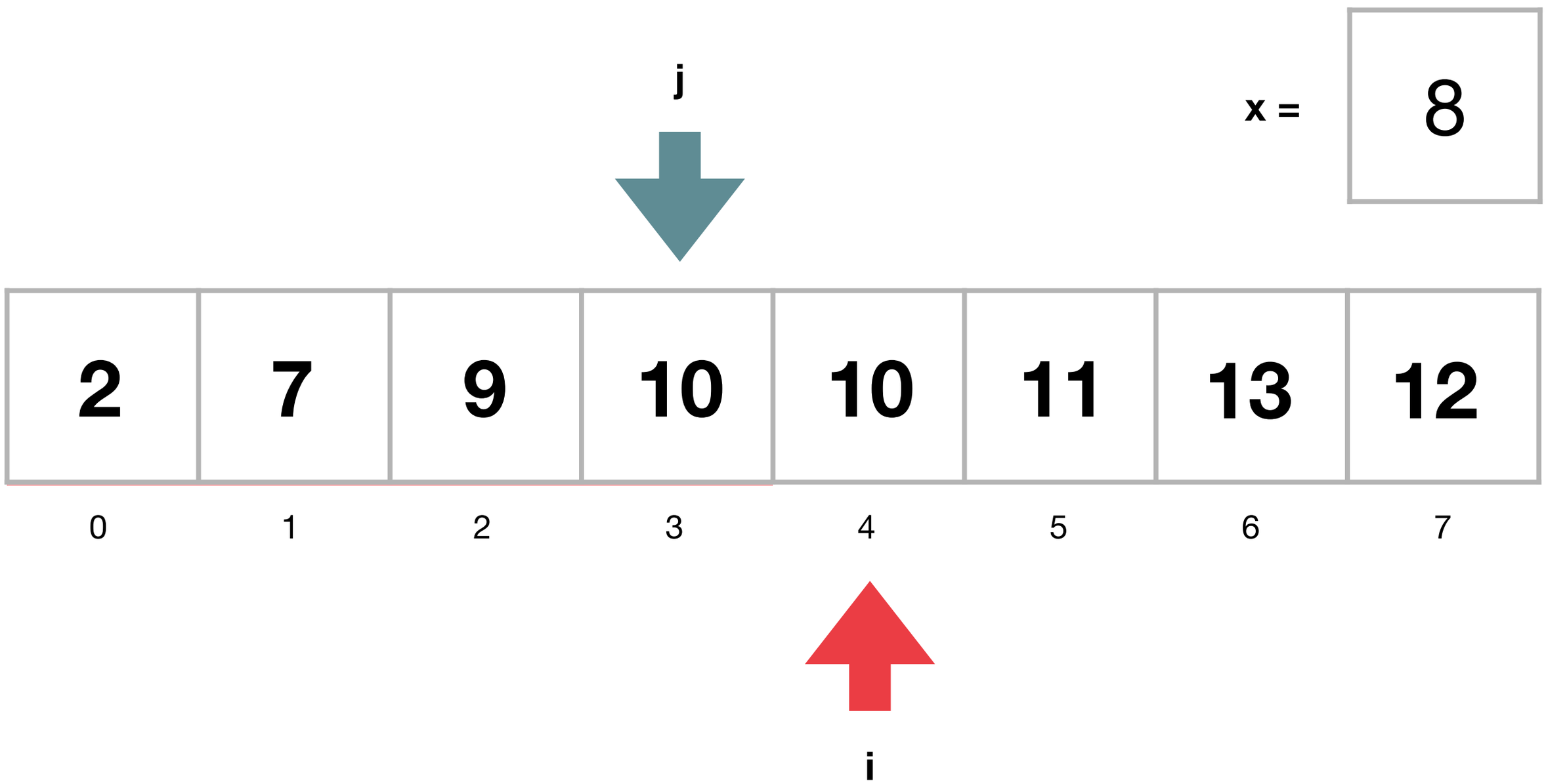
Tri par insertion



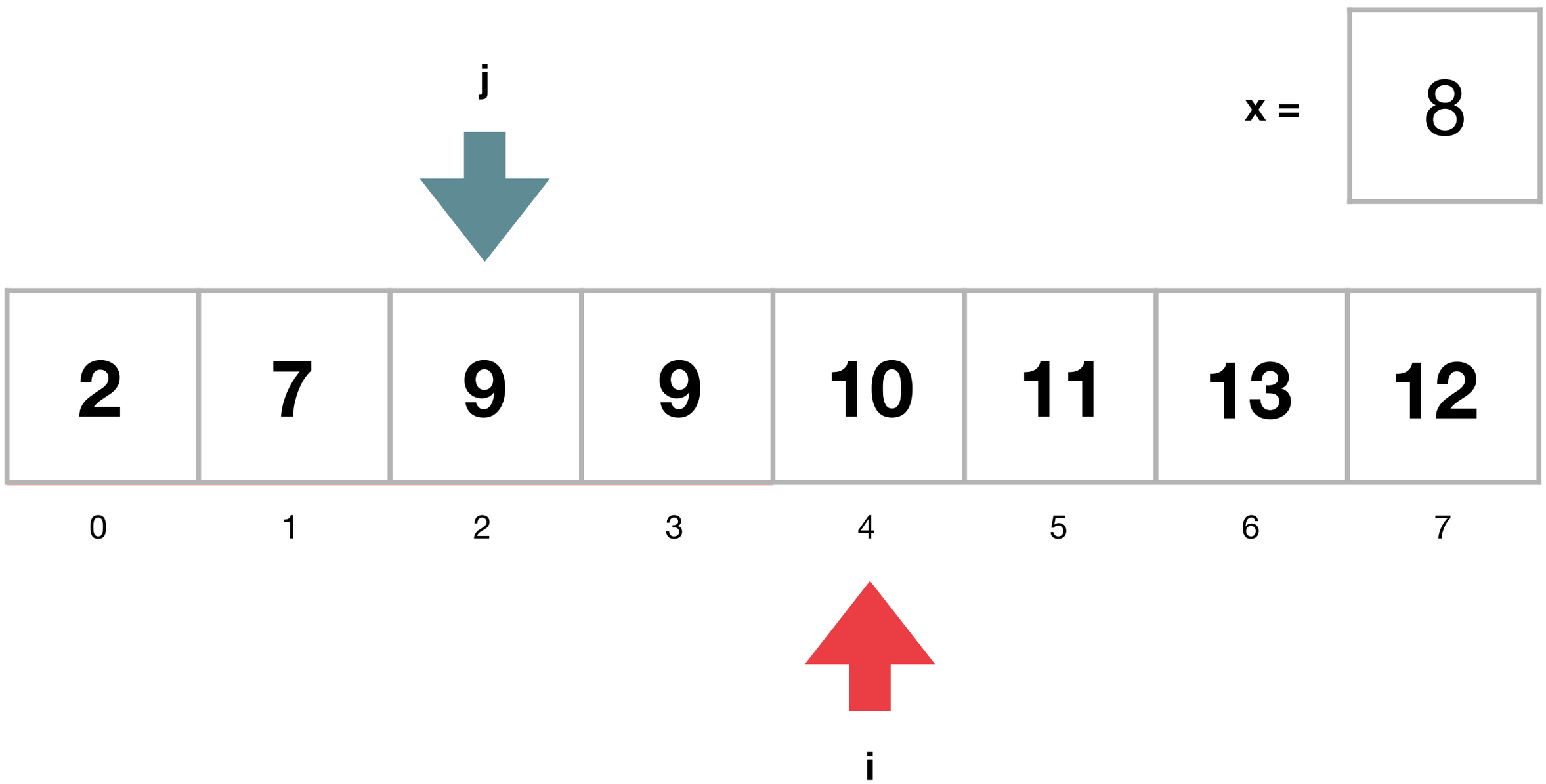
Tri par insertion



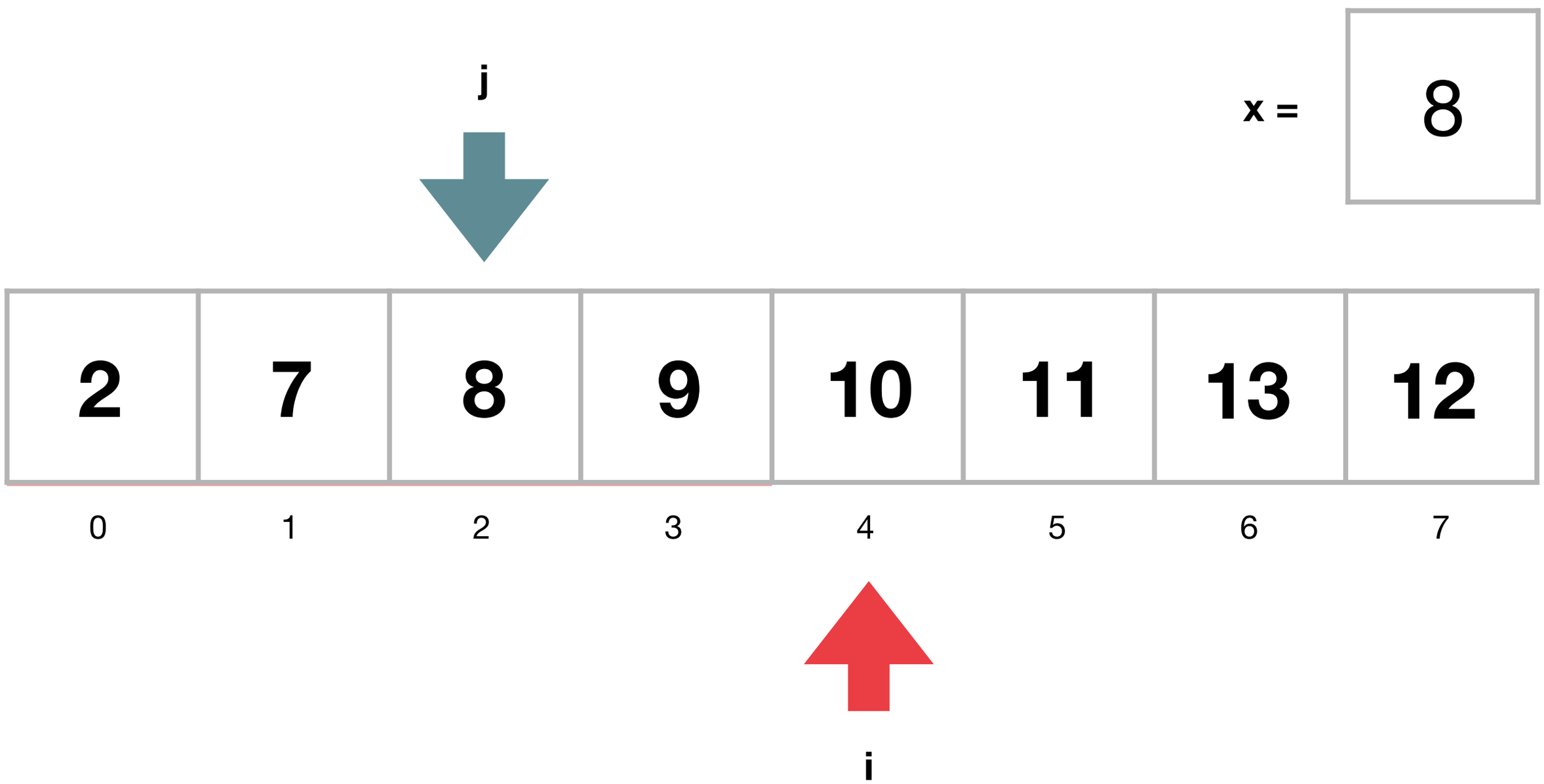
Tri par insertion



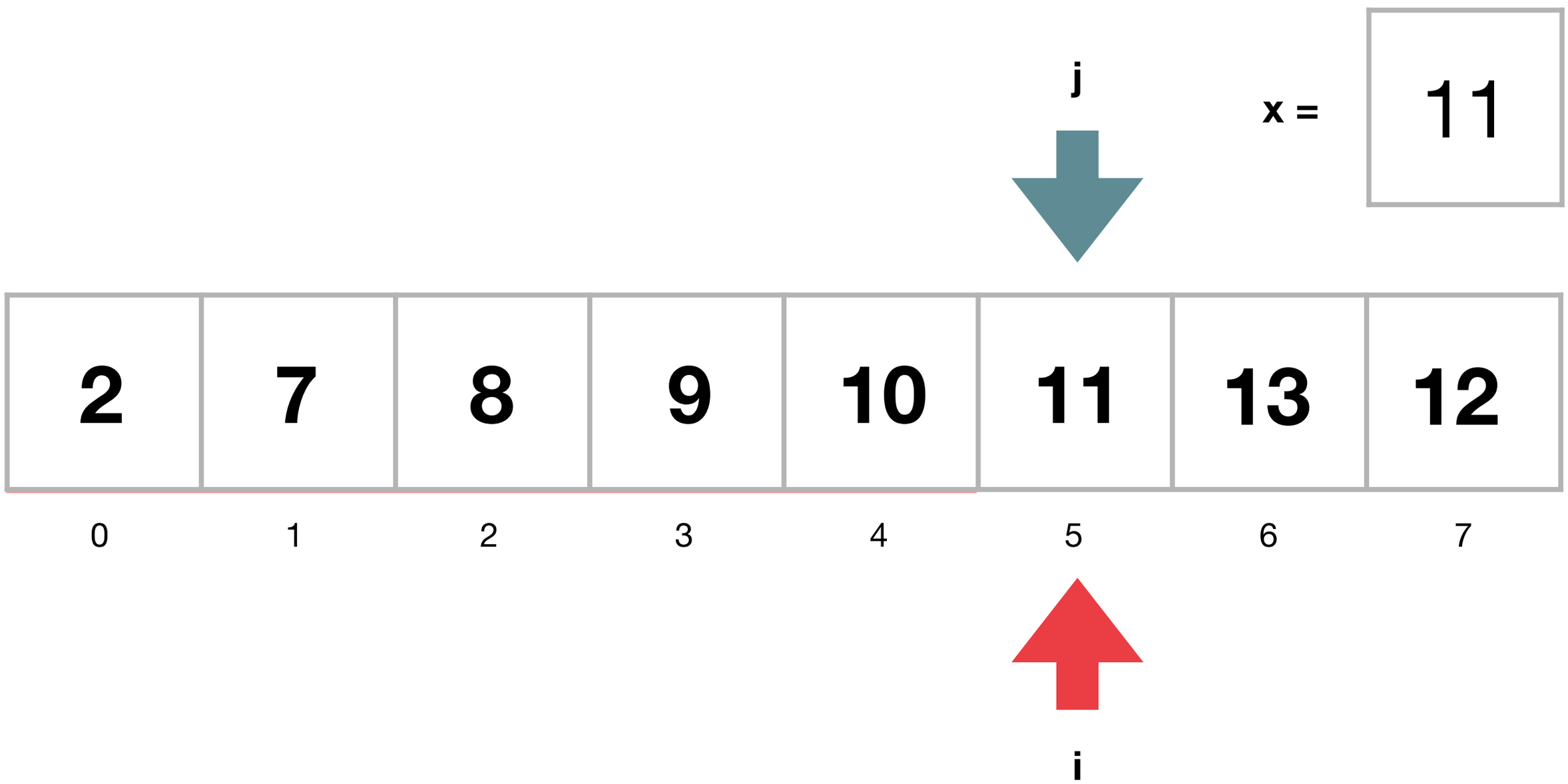
Tri par insertion



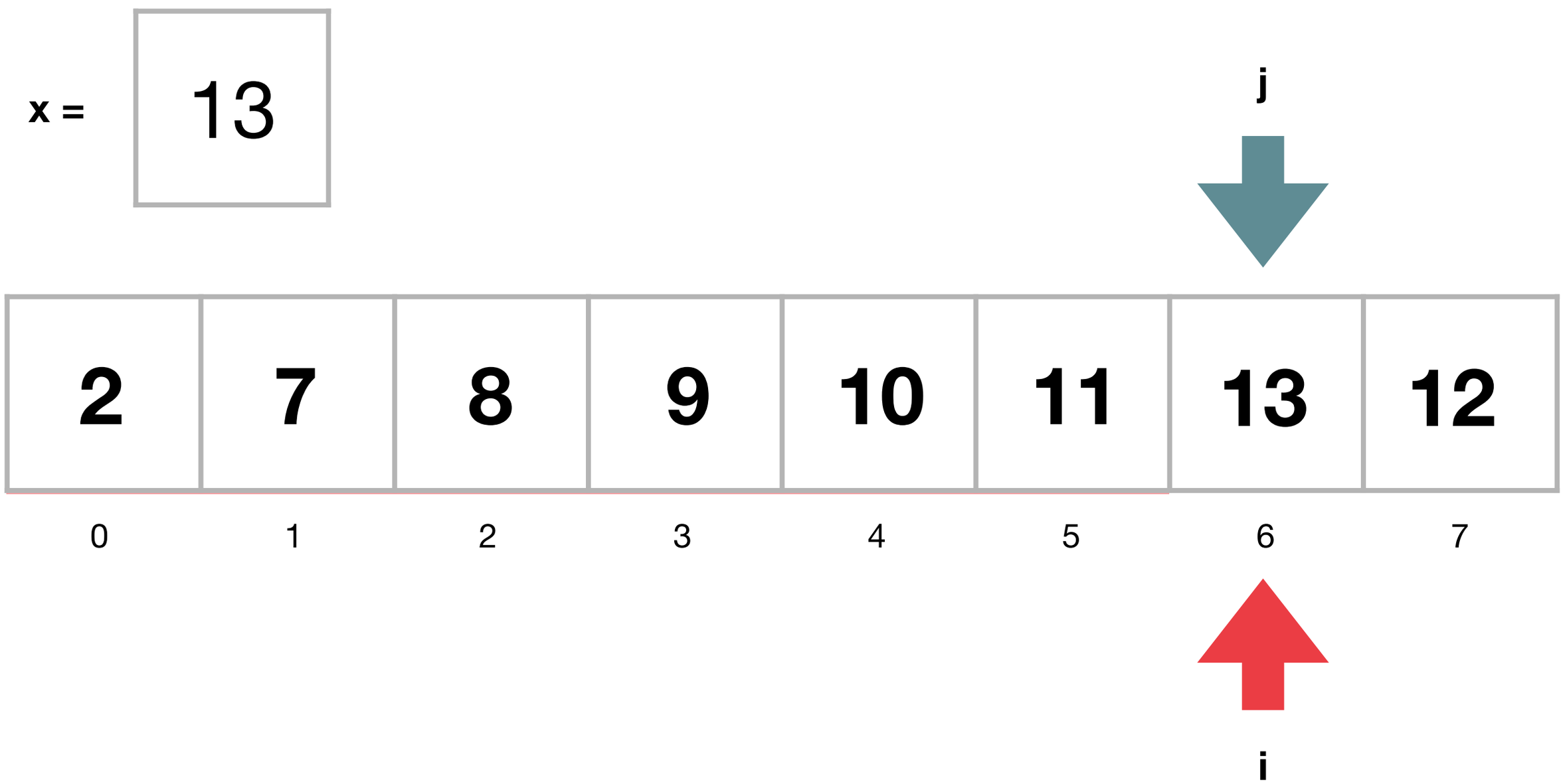
Tri par insertion



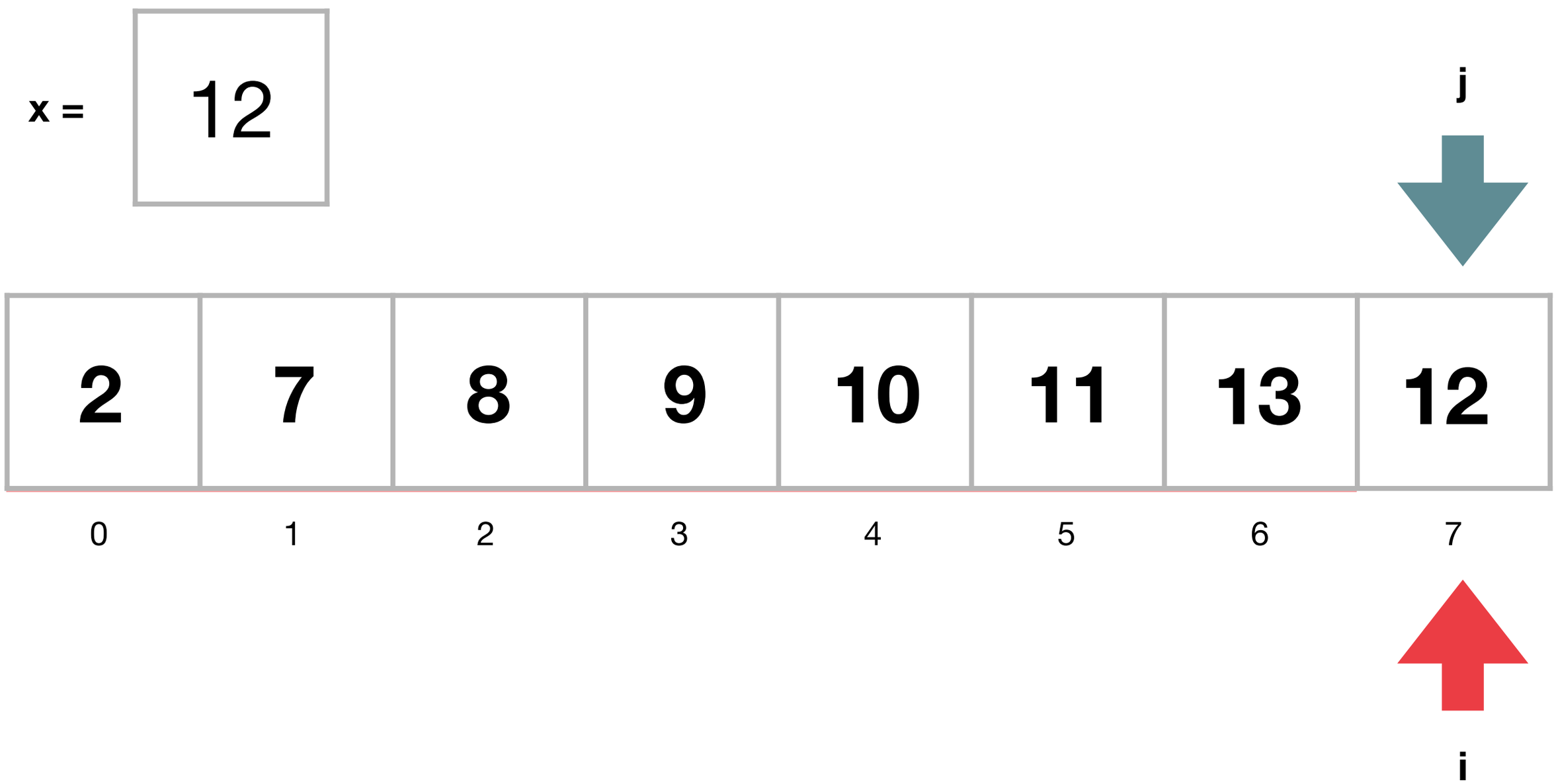
Tri par insertion



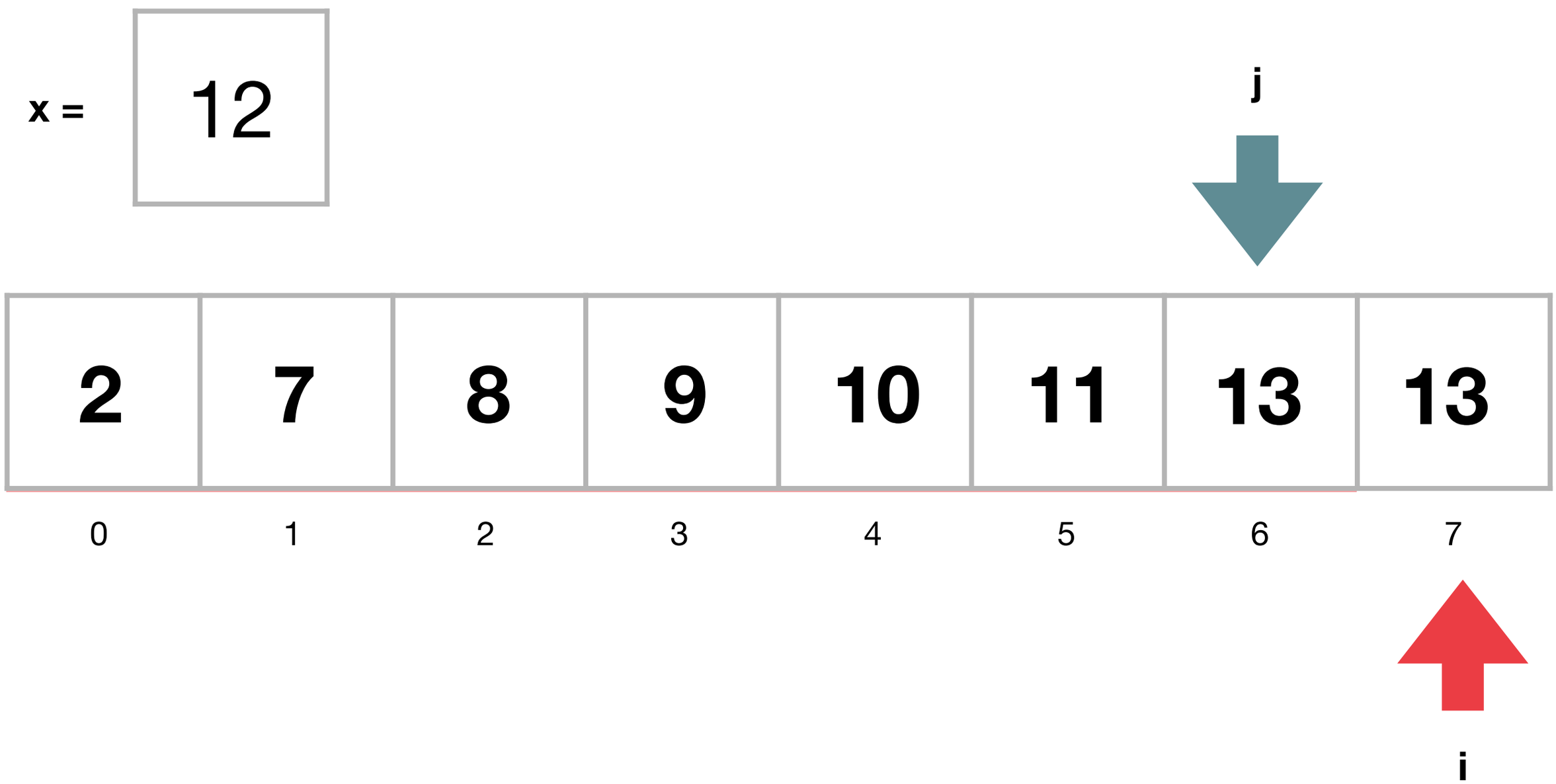
Tri par insertion



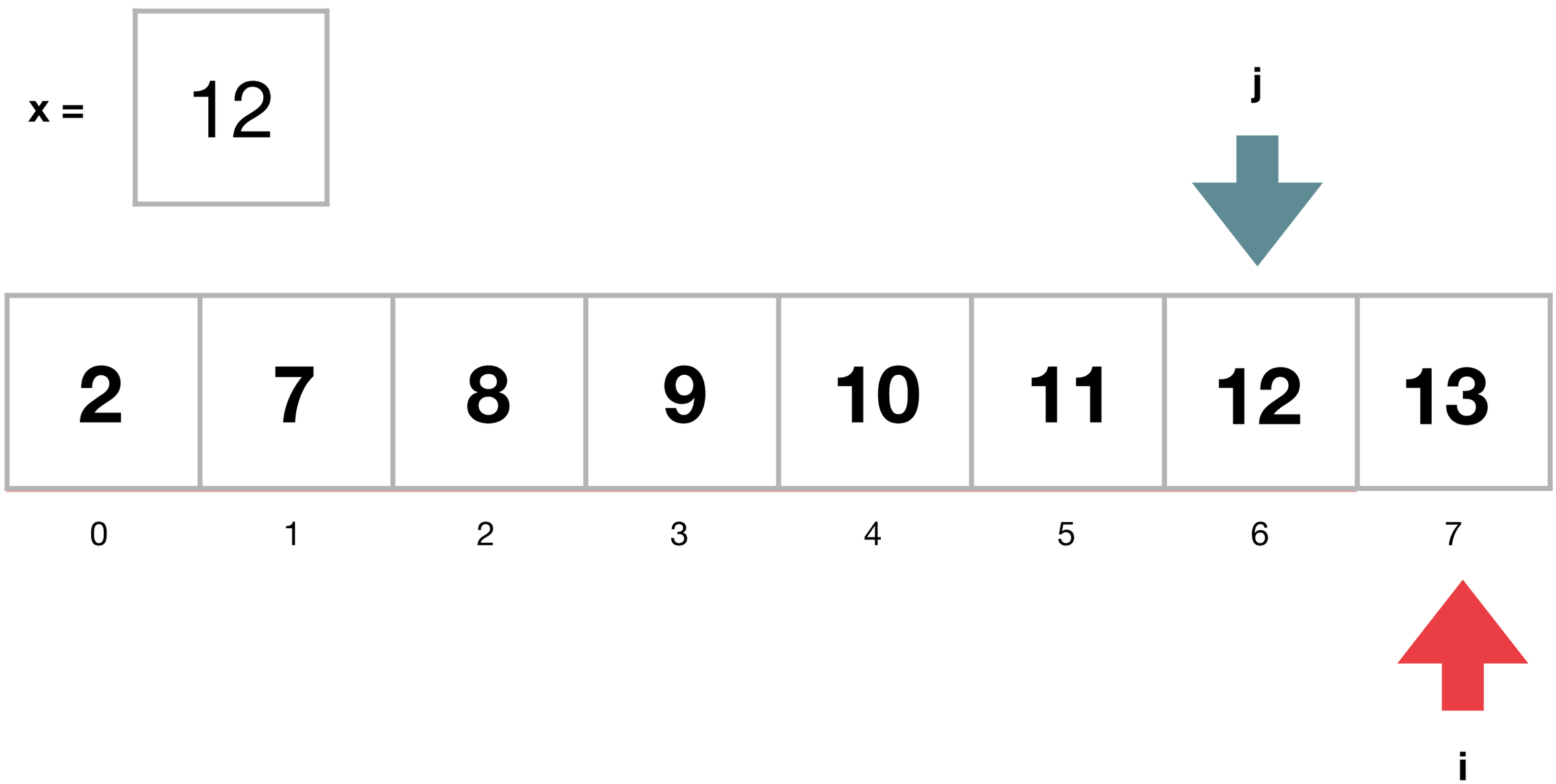
Tri par insertion



Tri par insertion



Tri par insertion



Tri par insertion

x = 12

j

2	7	8	9	10	11	12	13
0	1	2	3	4	5	6	7

i

Terminaison

```
procedure trier-par-insertion(T)
    n ≔ longueur(T)
    pour i ≔ 1 à n – 1 faire (on saute le 0)
        x ≔ T[i]
        j ≔ i
        tant que j > 0 et x < T[j – 1] faire
            (décaler d'un élément)
            T[j] ≔ T[j – 1]
            j ≔ j – 1
        fin tant que
        (ici x ≥ T[j – 1] ou bien j = 0)
        T[j] ≔ x
    fin pour
fin procedure
```

- La boucle **pour** termine toujours
- La boucle **tant que** termine (au pire) quand j = 0

Correction

```
procedure trier-par-insertion(T)
    n ≔ longueur(T)
    pour i ≔ 1 à n – 1 faire (on saute le 0)
        x ≔ T[i]
        j ≔ i
        tant que j > 0 et x < T[j – 1] faire
            (décaler d'un élément)
            T[j] ≔ T[j – 1]
            j ≔ j – 1
        fin tant que
        (ici x ≥ T[j – 1] ou bien j = 0)
        T[j] ≔ x
    fin pour
fin procedure
```

- Le sous-tableau T[0, …, i – 1] est trié au début de la boucle **pour**

Efficacité

```
procedure trier-par-insertion(T)
    n ≔ longueur(T)
    pour i ≔ 1 à n – 1 faire (on saute le 0)
        x ≔ T[i]
        j ≔ i
        tant que j > 0 et x < T[j – 1] faire
            (décaler d'un élément)
            T[j] ≔ T[j – 1]
            j ≔ j – 1
        fin tant que
        (ici x ≥ T[j – 1] ou bien j = 0)
        T[j] ≔ x
    fin pour
fin procedure
```

- $O(n)$ opérations dans le meilleur des cas
- $O(n^2)$ opérations dans le pire des cas