TP N°3: L'ALGORITHME DE TRI RAPIDE

PRÉPARER PAR

IDRISS ABDELMAGHNI KEBLADJ.
BOUCHACHI MEHDI MOHAMED.

RESPONSABLE DU TP:

MME S.AROUSSI

Visual graph tree

✤ENVIRONNEMENT UTILISÉ

👉 **Processeur (CPU)**	**I7 5eme Gen**	*Assure la rapidité et la stabilité lors du développement et de la compilation.*
👉 **Mémoire vive (RAM)**	**16 GB DDR4**	*Permet d'exécuter plusieurs outils et serveurs sans ralentissement.*
👉 **système d'exploitation**	Windows 10	*Fournit un environnement fiable et compatible pour le développement web.*
👉 **Langage de programmation**	JavaScript	*Utilisé pour créer des interfaces interactives et dynamiques.*
👉 **Environnement**	Visual Studio Code	*Outil principal d'édition et de gestion du code source.*
👉 **Bibliothèques**	**React**	*Ensemble de modules facilitant la création d'applications réactives.*
👉 **Other tools**	**React icons/ React hot toast/ Styled-components/ Assistant Utilisée, ChatGPT 6.8%**	

✤PLAN

👉L’ algorithme de tri rapide

- ✤ Introduction.
- ✤ Définition.
- ✤ Principe de fonctionnement.
- ✤ Implémentation.
- ✤ complexité.

✤ INTRODUCTION.

👉 Un algorithme de tri permet d'organiser une collection d'objets selon un ordre déterminé. Cette collection doit être munie d'une relation d’ordre. Il est possible de définir un algorithme de tri indépendamment de la fonction d’ordre utilisée.

✤ DÉFINITION.

👉 **Le tri rapide** (en anglais quick sort) est **un algorithme de tri** invente par C.A.R. Hoare en 1961 et fonde sur la méthode de conception **diviser pour régner**. Il est généralement **utilise sur des tableaux**, mais peut aussi être adapte aux **autres types de données**.

✤ PRINCIPE DE FONCTIONNEMENT.

👉 Fonctionnalités

L'algorithme peut s'effectuer récursivement :

- 👉Le premier élément de la liste a trier est appelé "pivot" .
- 👉 On créé un premier sous-ensemble constitue des éléments plus petits que le pivot. on les place à gauche du pivot dans l'ordre ou ils sont dans la liste à trier.
- 👉 On créer de la même façon un deuxième sous-ensemble constitué des éléments plus grands que le pivot. on les place à droite de celui-ci dans L'ordre ou ils apparaissent dans le tableau.
- 👉 On procède de meme par récursivité sur les deux sous-ensembles jusqu'à obtenir des sous ensembles d'un seul élément.

✤ PRINCIPE DE FONCTIONNEMENT.

👉 Importance du Choix du Pivot

- Le choix du pivot est CRUCIAL pour les performances de Quicksort. Un mauvais choix peut dégrader les performances de O(n log n) à $O(n^2)$.

👉 Stratégies de Choix du Pivot

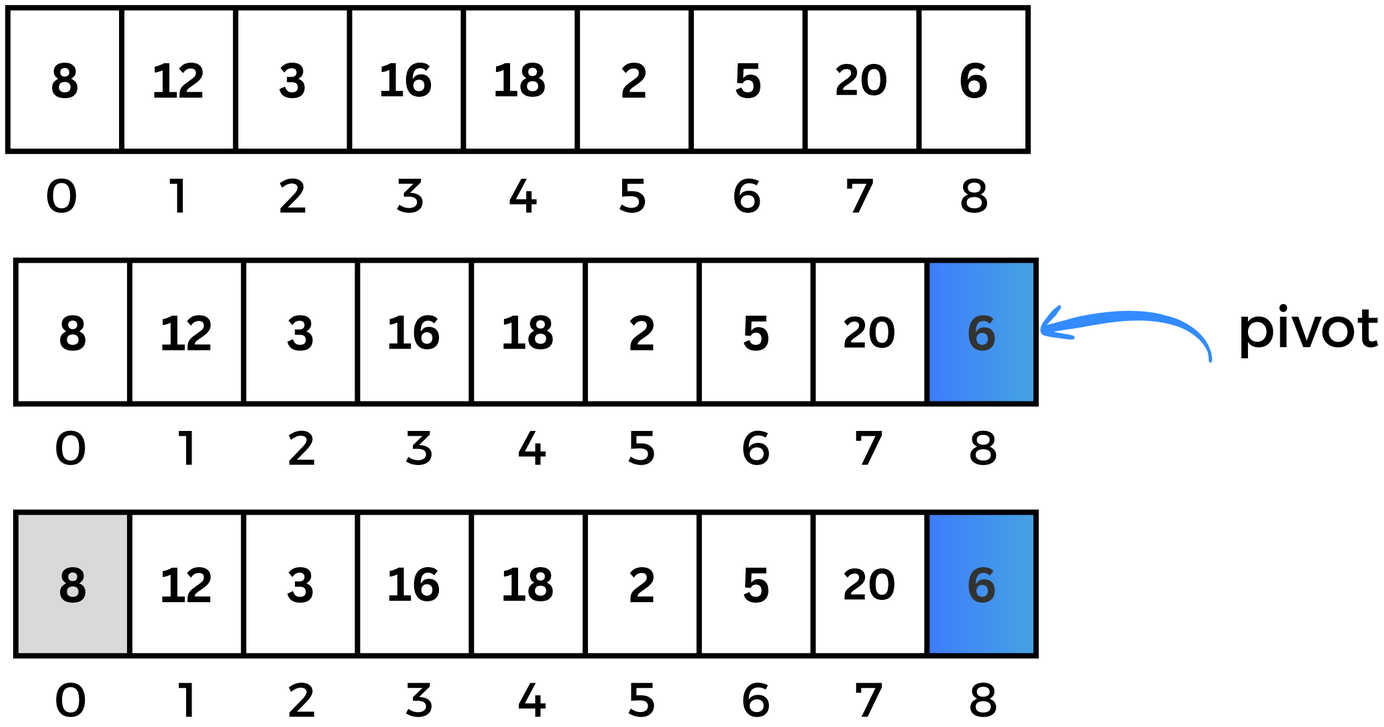
1. Première élément.
2. Dernier élément
3. Pivot aléatoire
4. Élément du milieu
5. Médiane de trois

✤ PRINCIPE DE FONCTIONNEMENT.

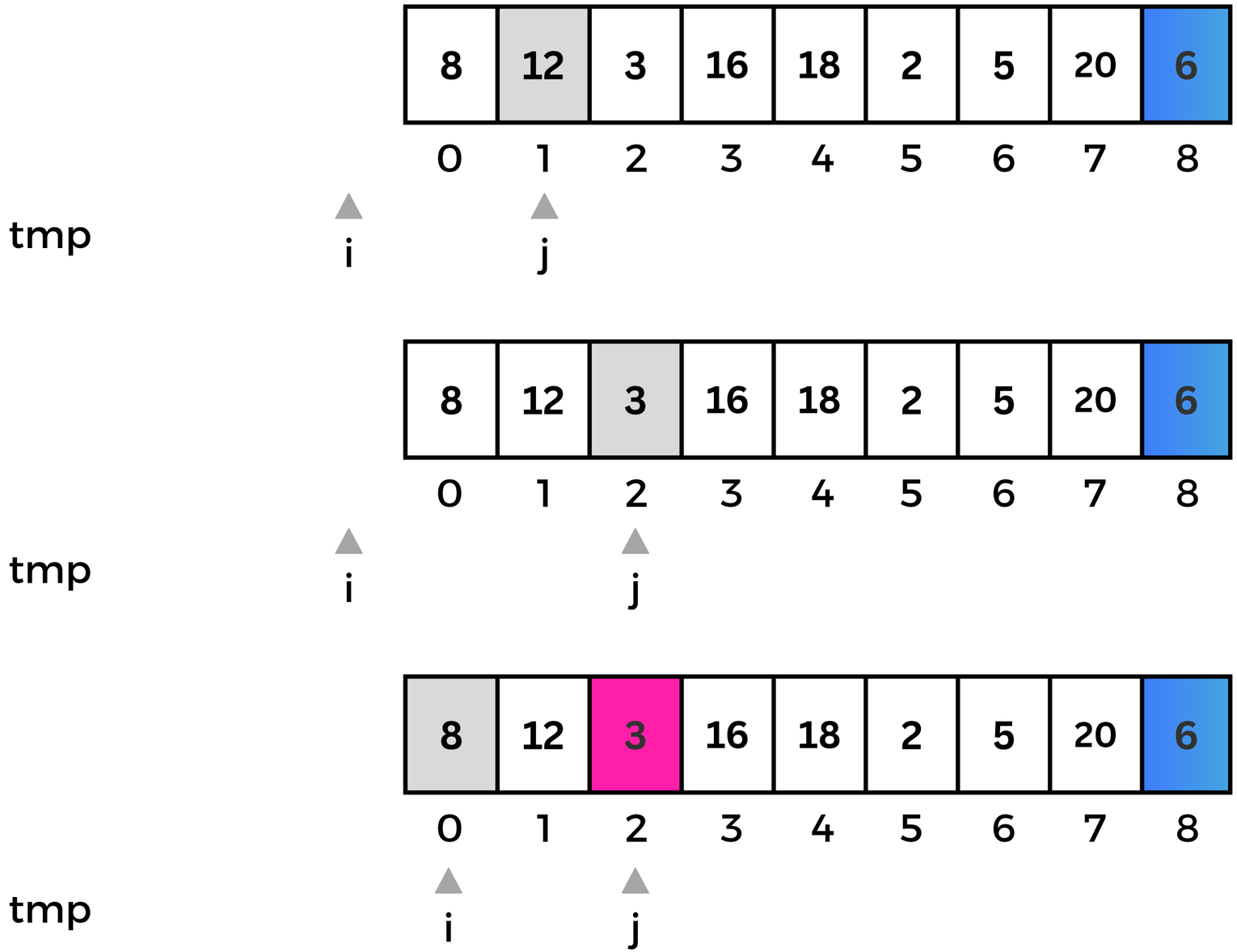
💡 Exemple 1:

- On va illustrer le tri rapide sur le tableau T=[8,12,3,16,18,2,5,20,6].
- On choisit le dernier élément comme pivot(pivot=6)

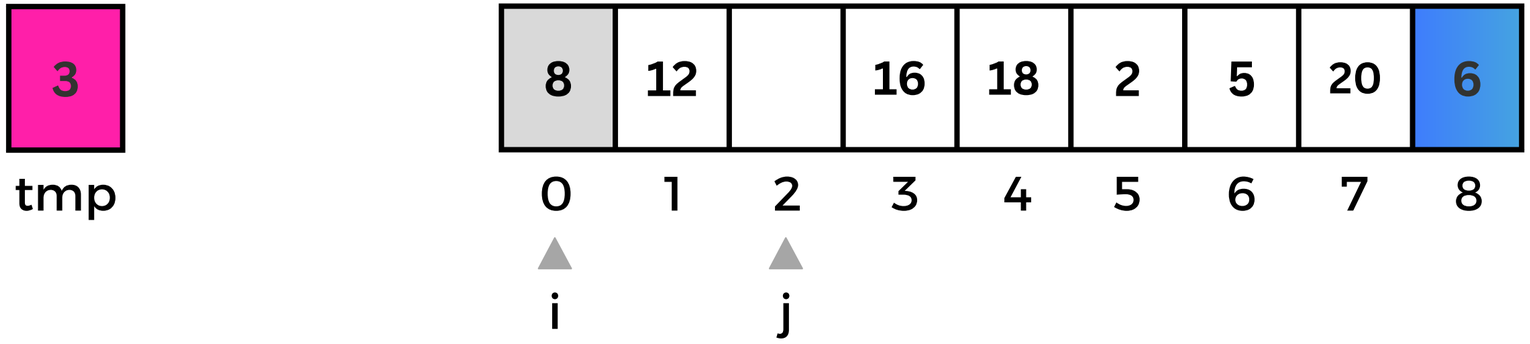
🔧 Étape 1 : Premier partitionnement

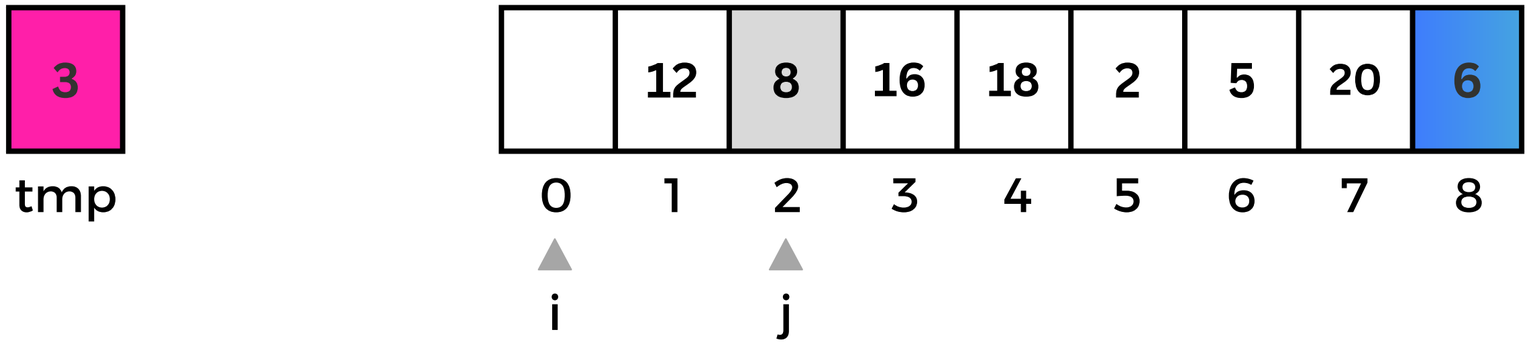


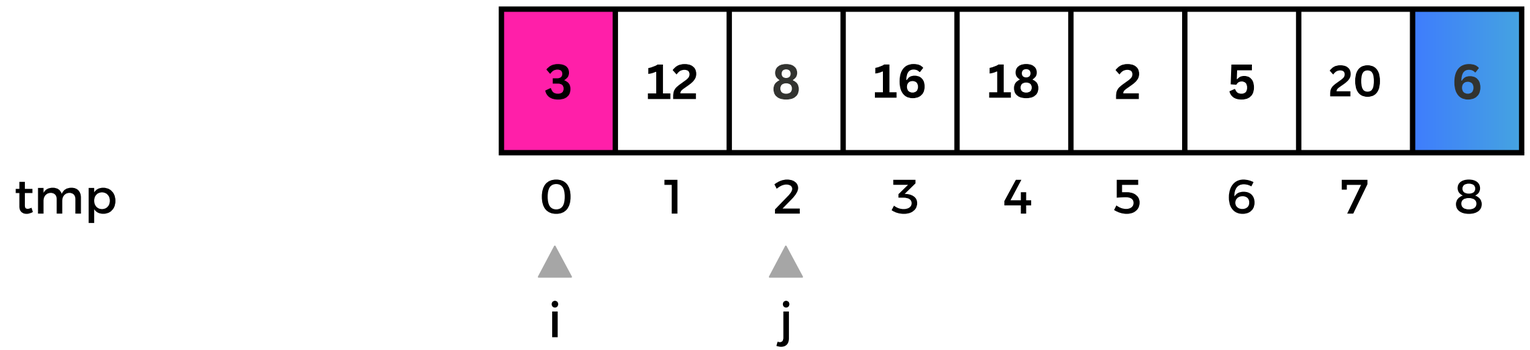
✤ PRINCIPE DE FONCTIONNEMENT.



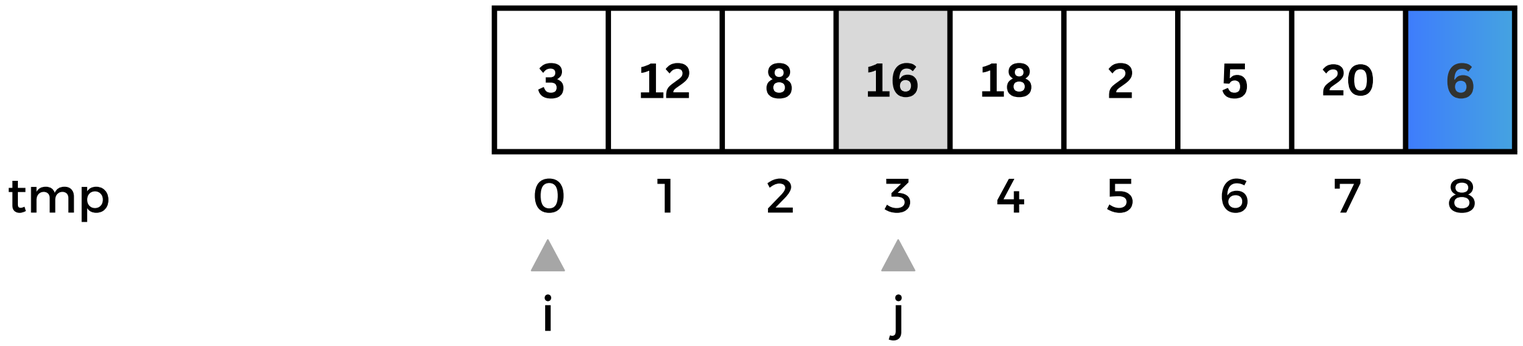
✤ PRINCIPE DE FONCTIONNEMENT.

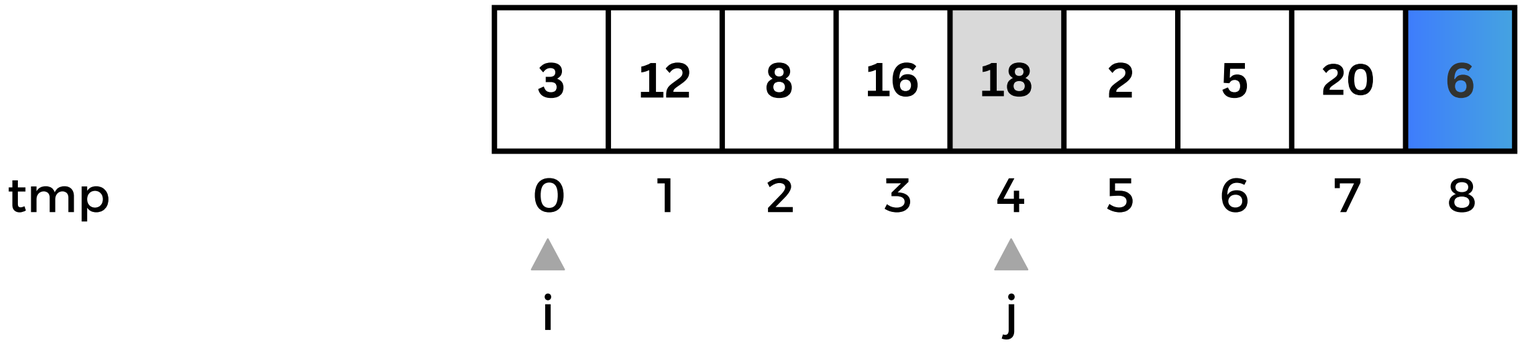


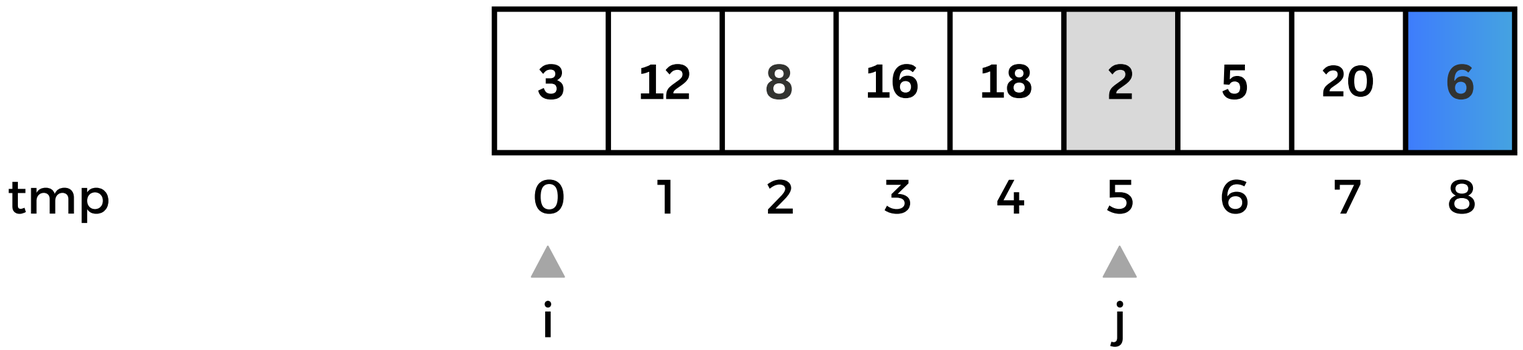




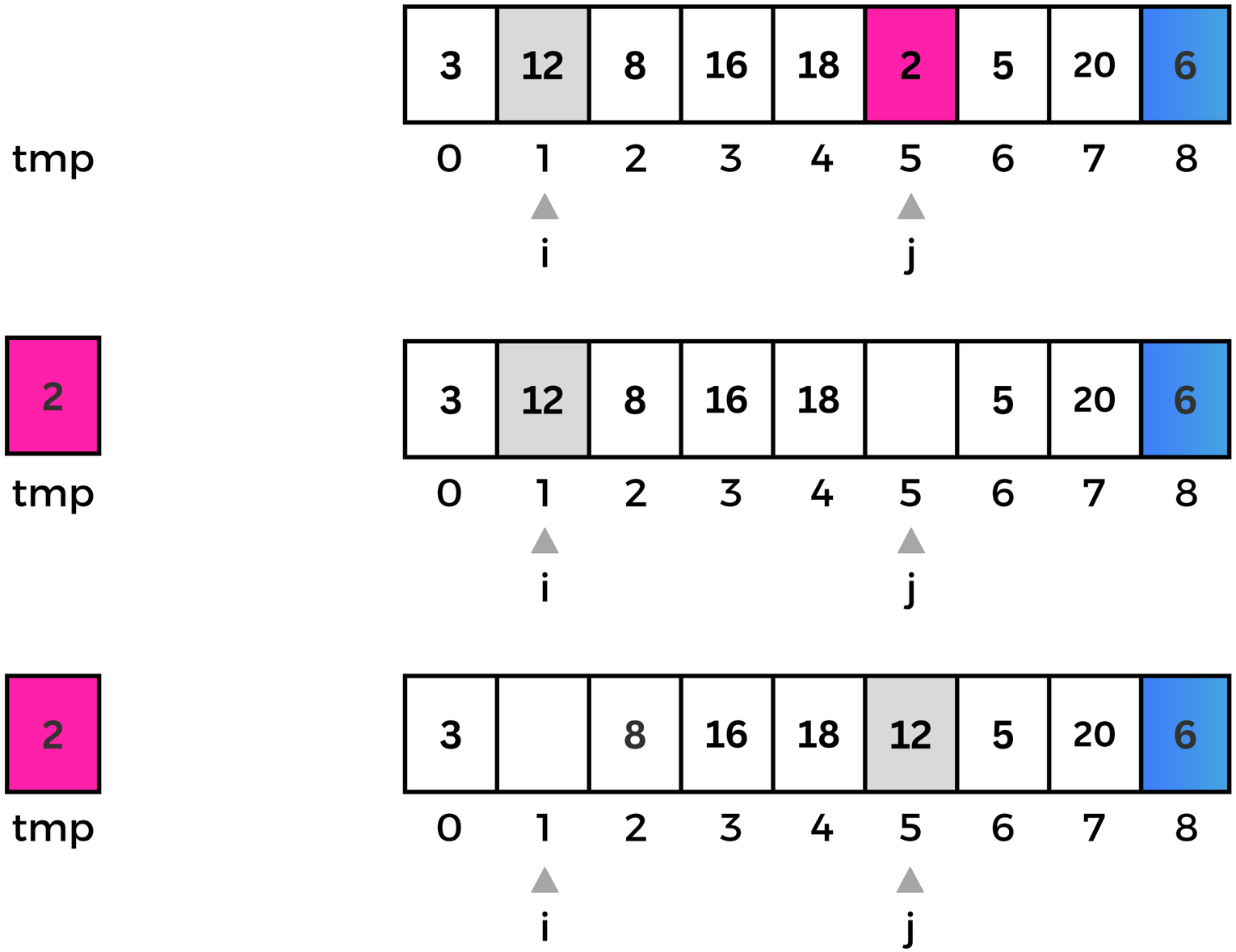
✤ PRINCIPE DE FONCTIONNEMENT.



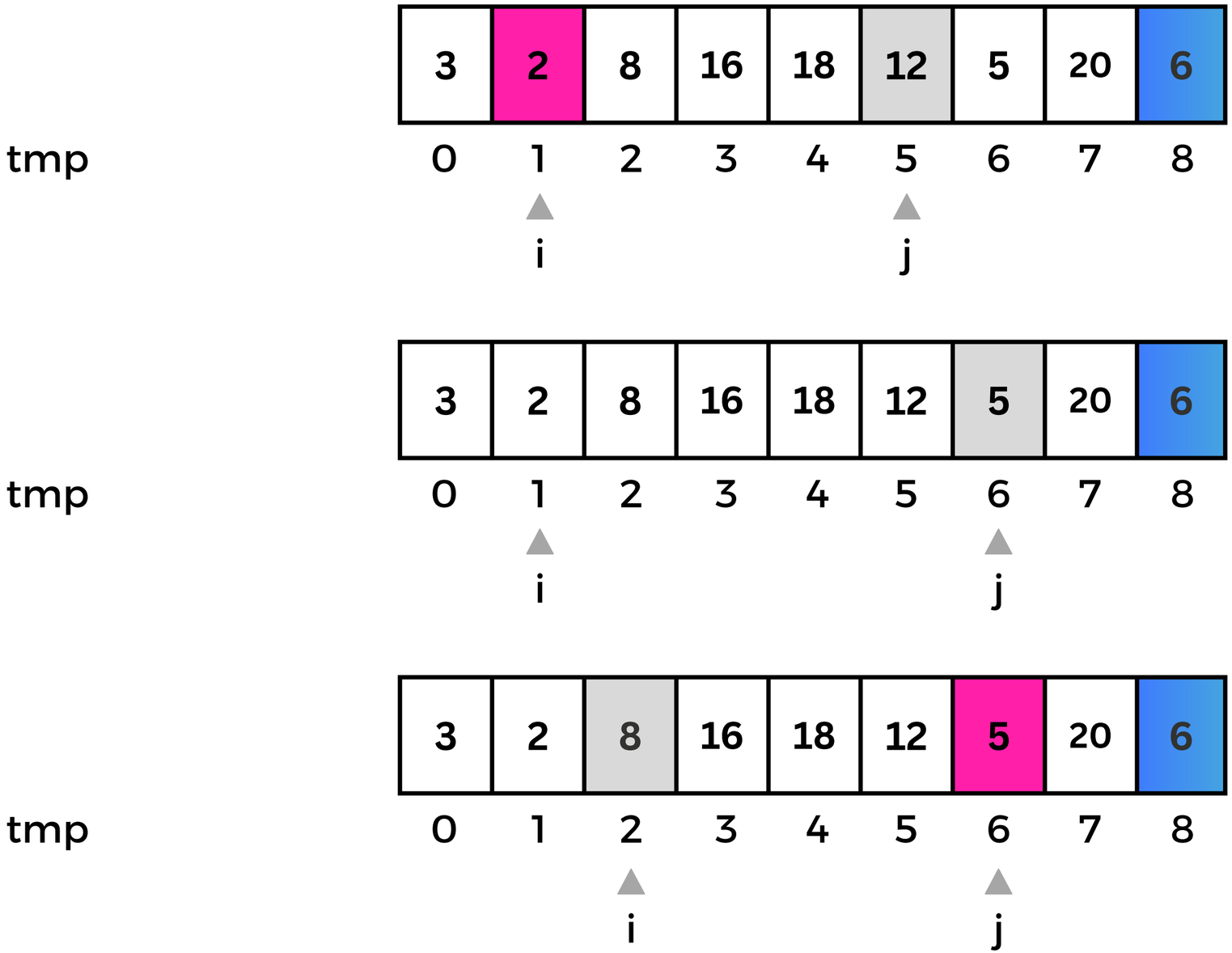




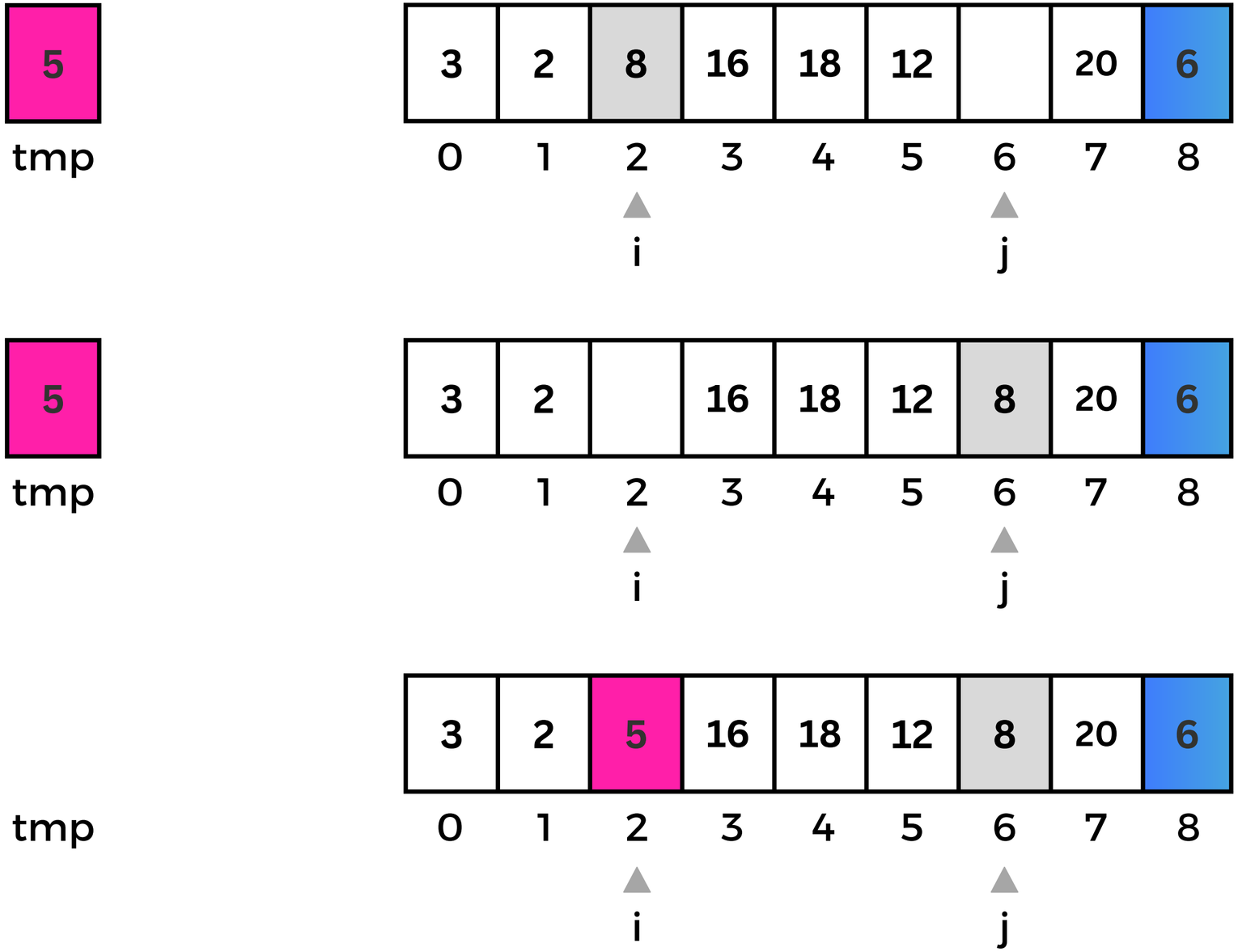
✤ PRINCIPE DE FONCTIONNEMENT.



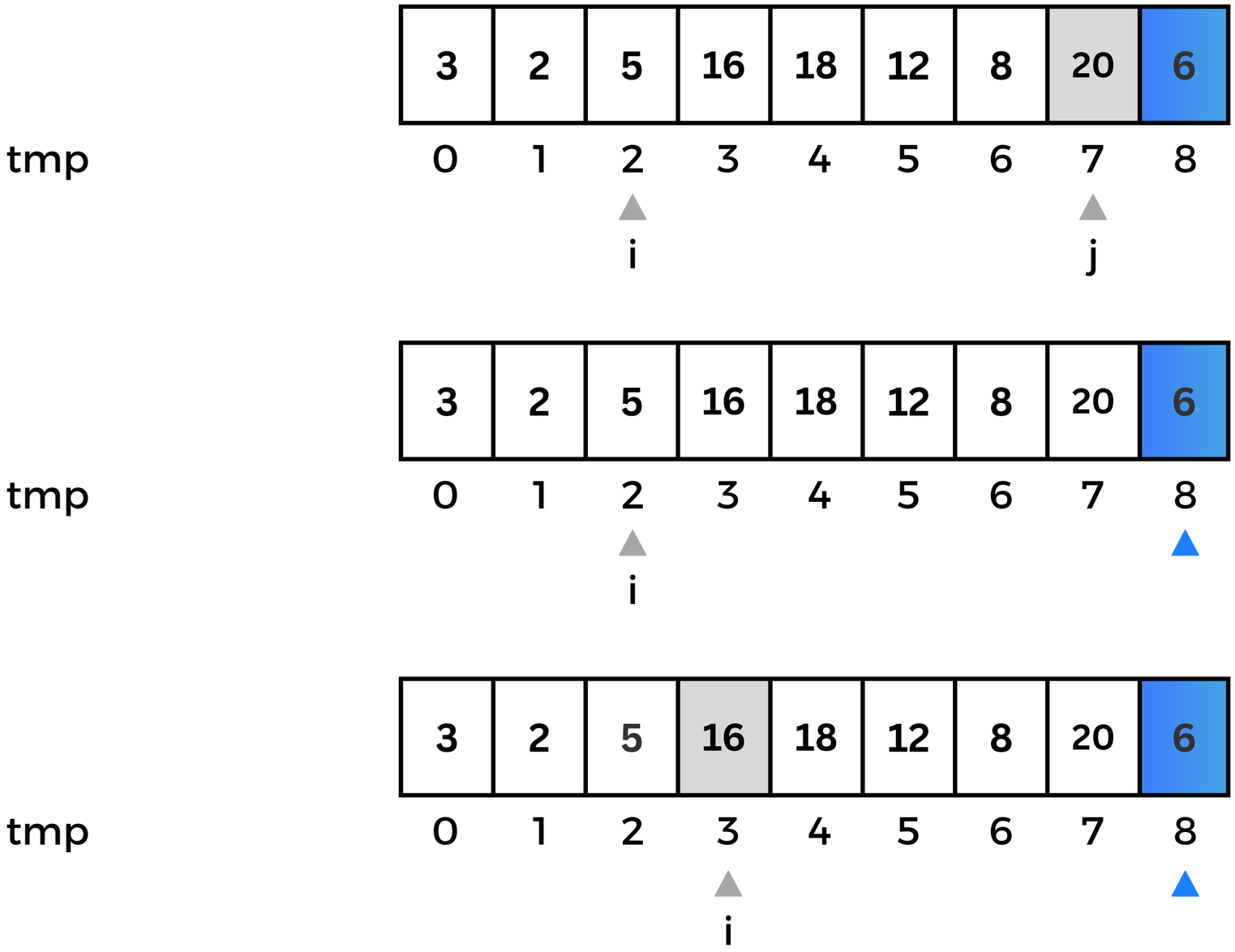
✤ PRINCIPE DE FONCTIONNEMENT.



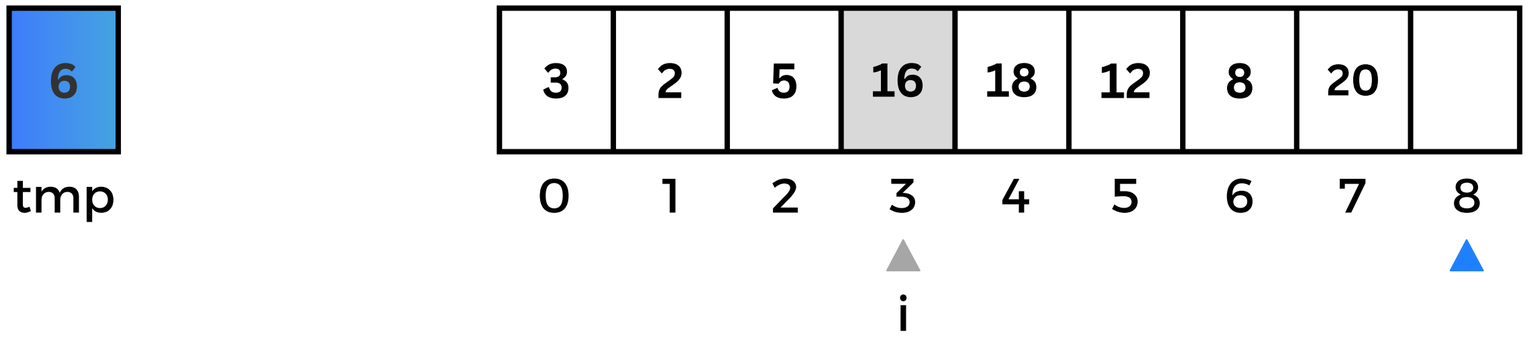
✤ PRINCIPE DE FONCTIONNEMENT.

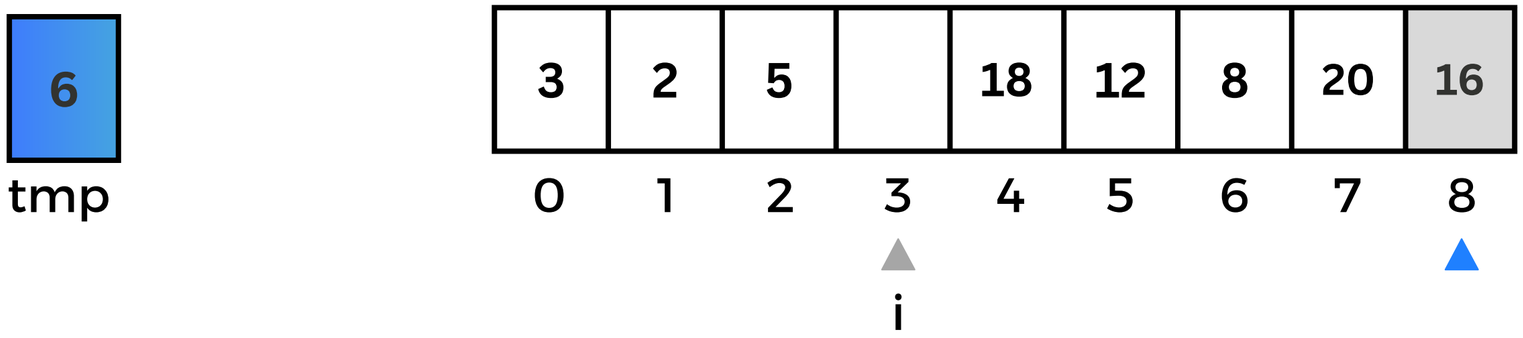


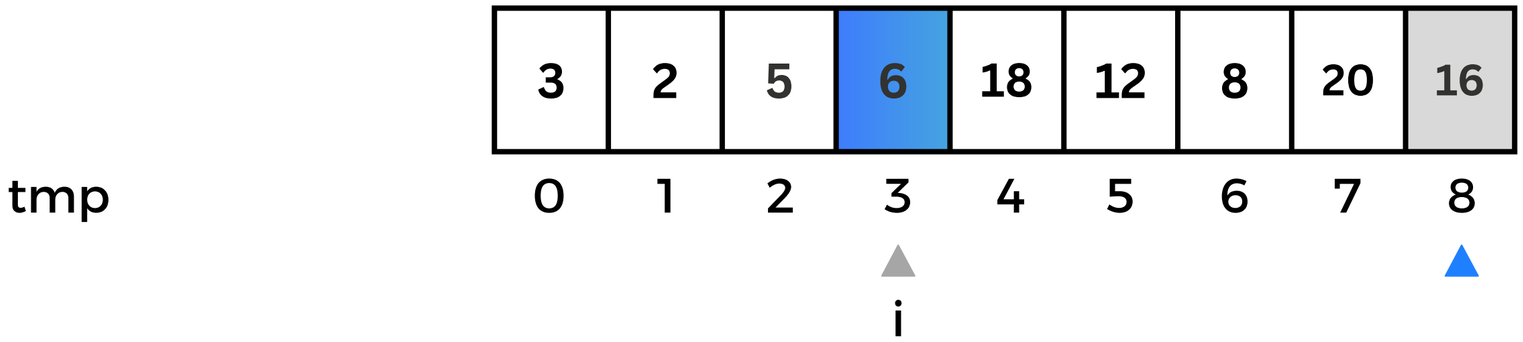
✤ PRINCIPE DE FONCTIONNEMENT.



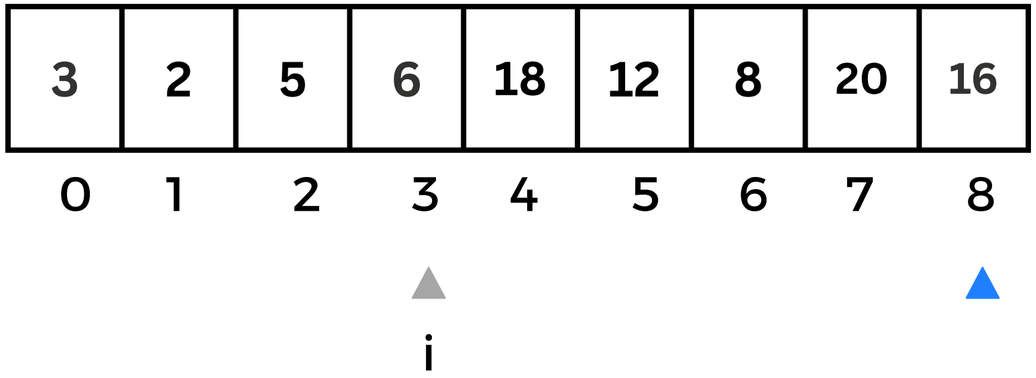
✤ PRINCIPE DE FONCTIONNEMENT.





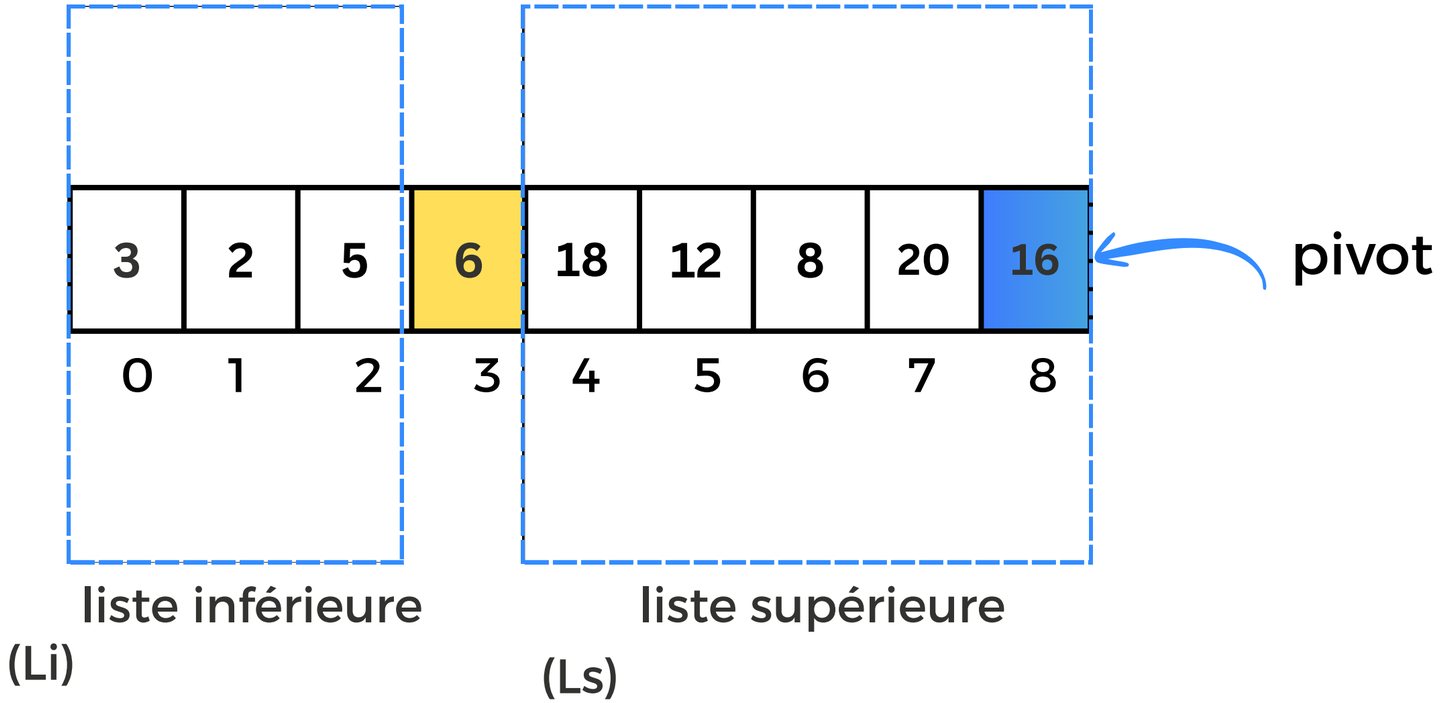


✤ PRINCIPE DE FONCTIONNEMENT.



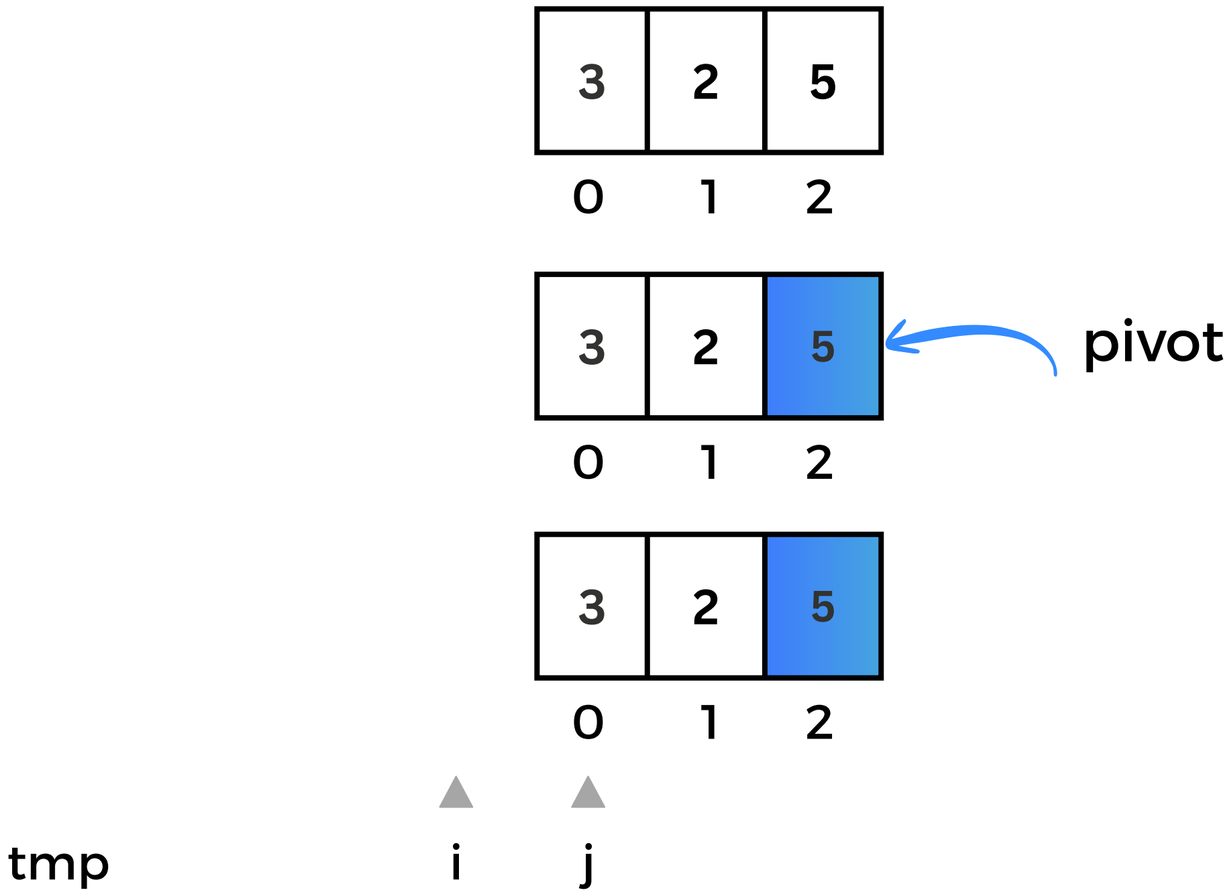
tmp

📊 **Résultat après partition 1 :**

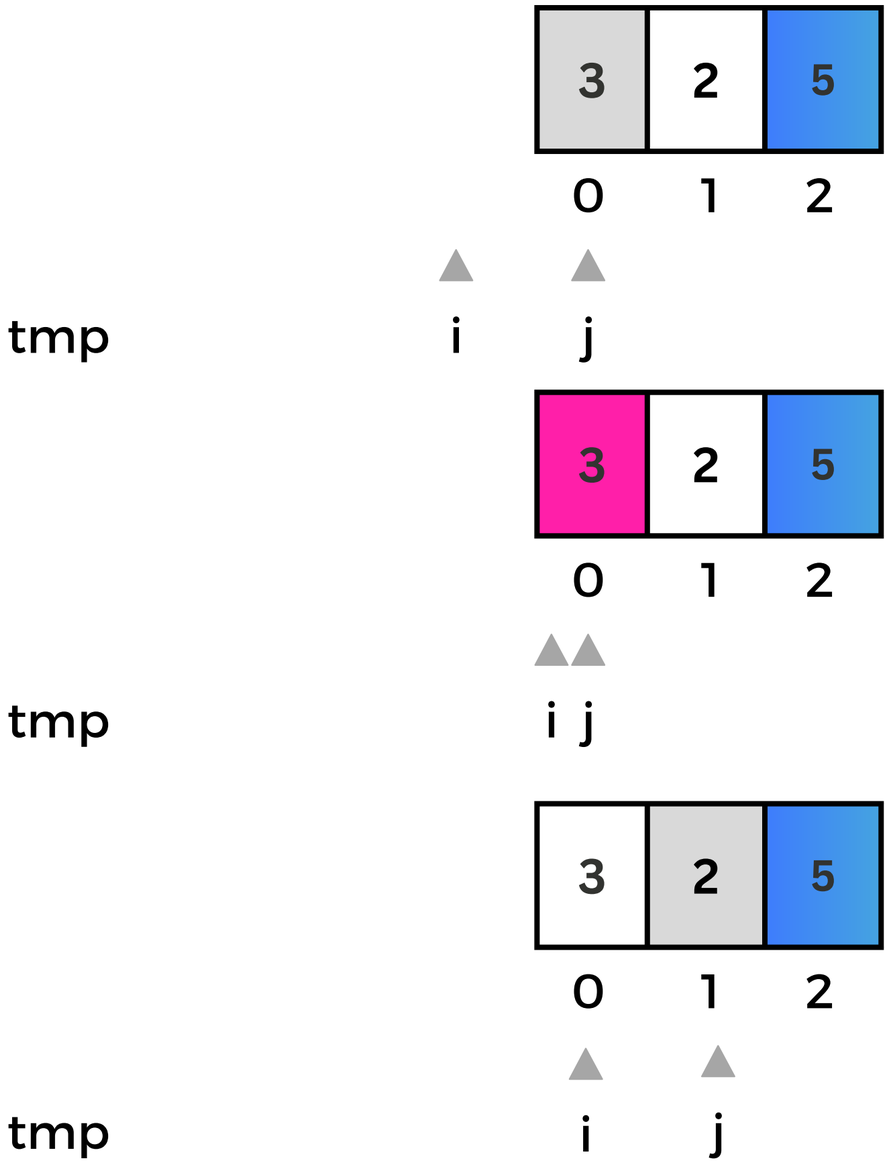


✤ PRINCIPE DE FONCTIONNEMENT.

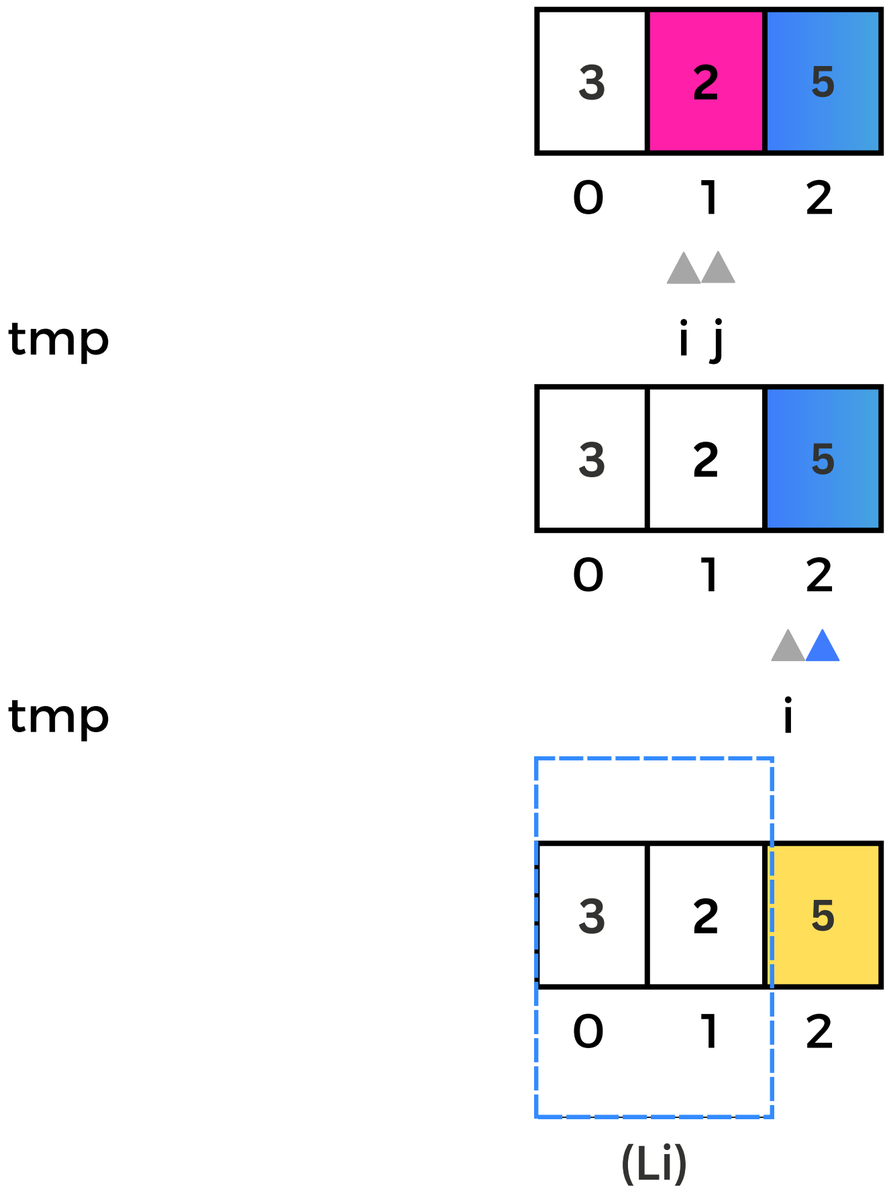
🔧 Étape 2.1 : Tri Li= [3, 2, 5]



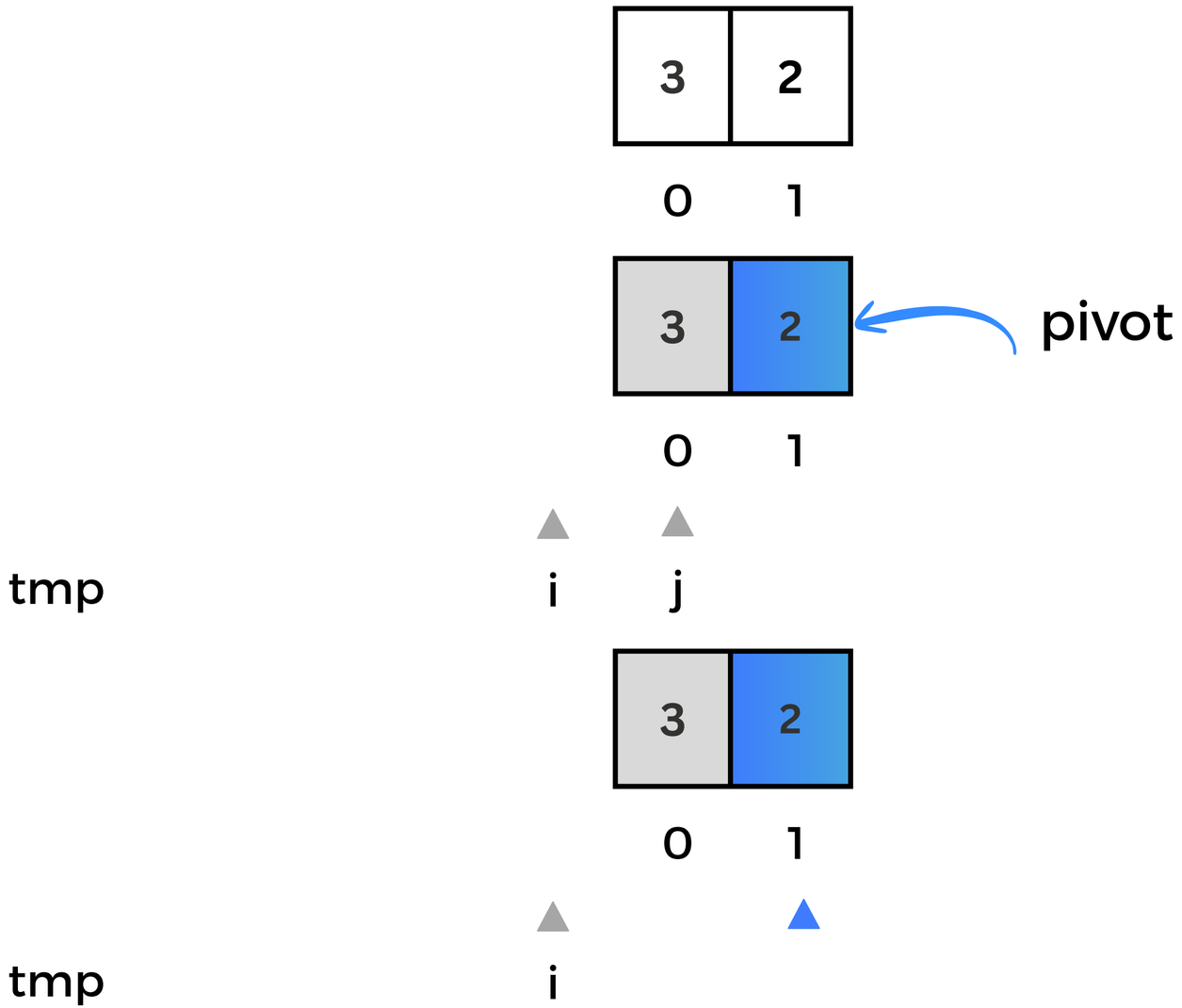
✤ PRINCIPE DE FONCTIONNEMENT.



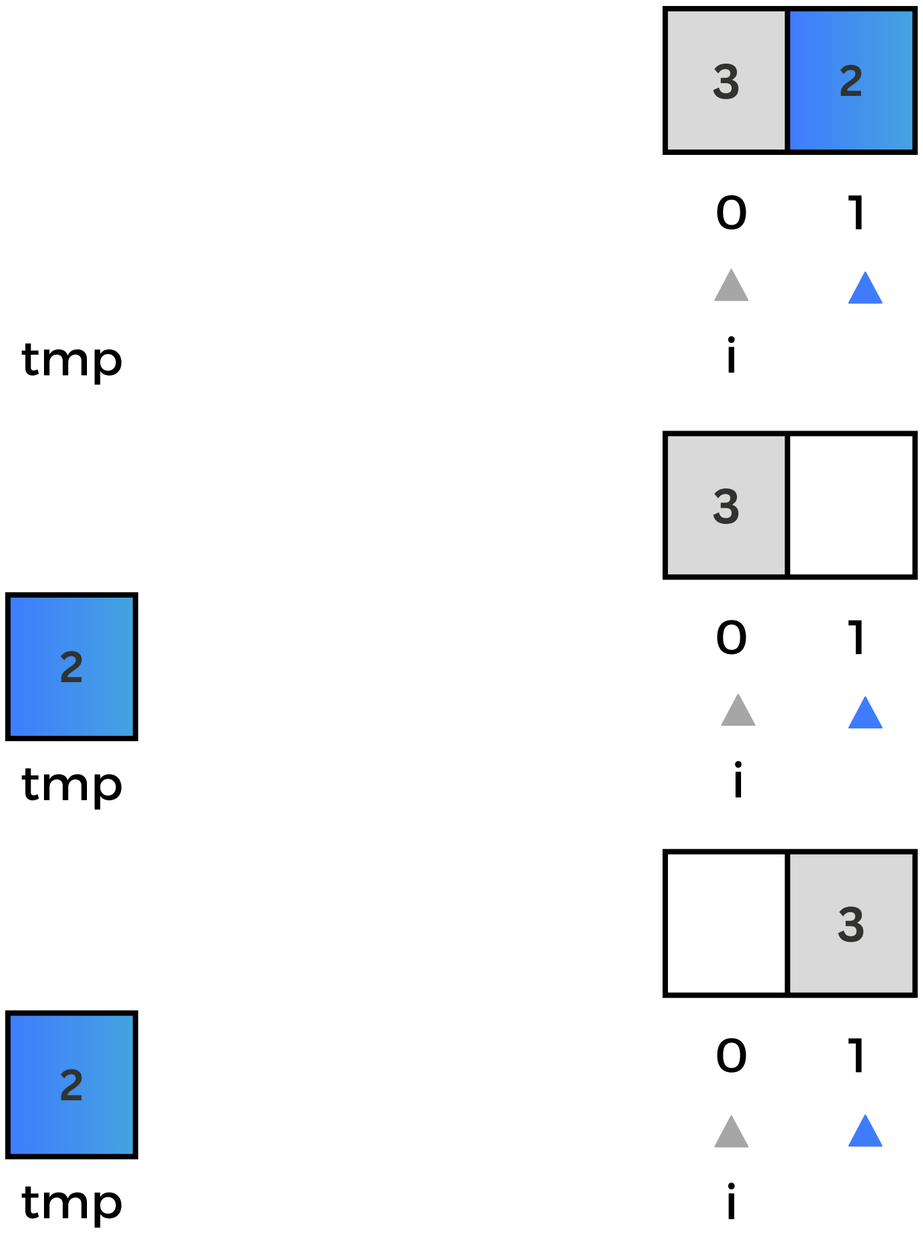
✤ PRINCIPE DE FONCTIONNEMENT.



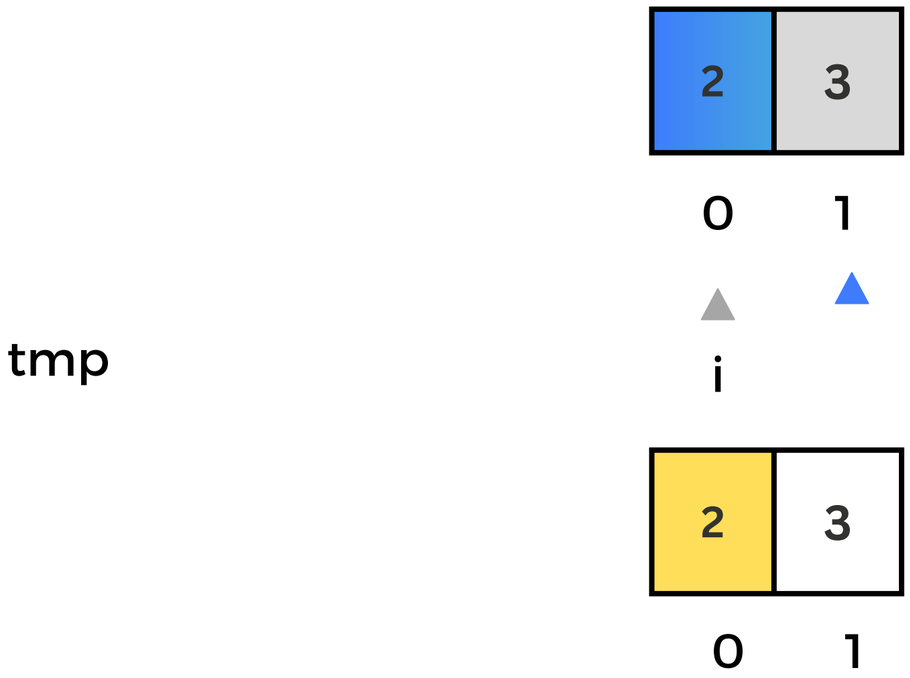
✤ PRINCIPE DE FONCTIONNEMENT.



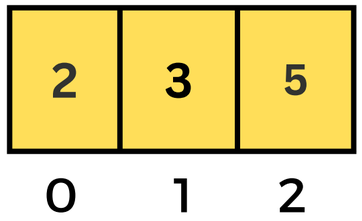
✤ PRINCIPE DE FONCTIONNEMENT.



✤ PRINCIPE DE FONCTIONNEMENT.

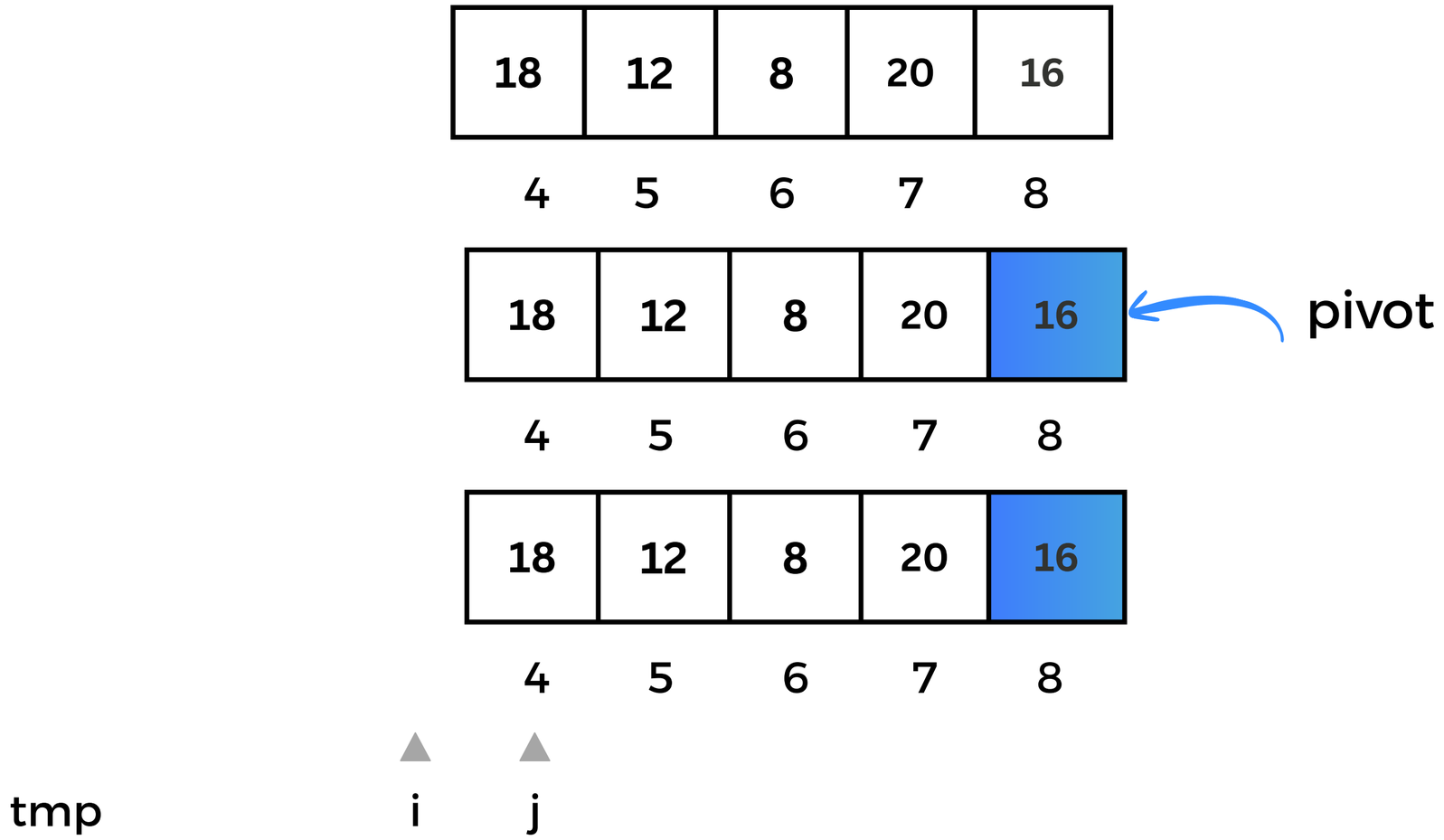


📊 **Résultat de l’étape 2.1: :**

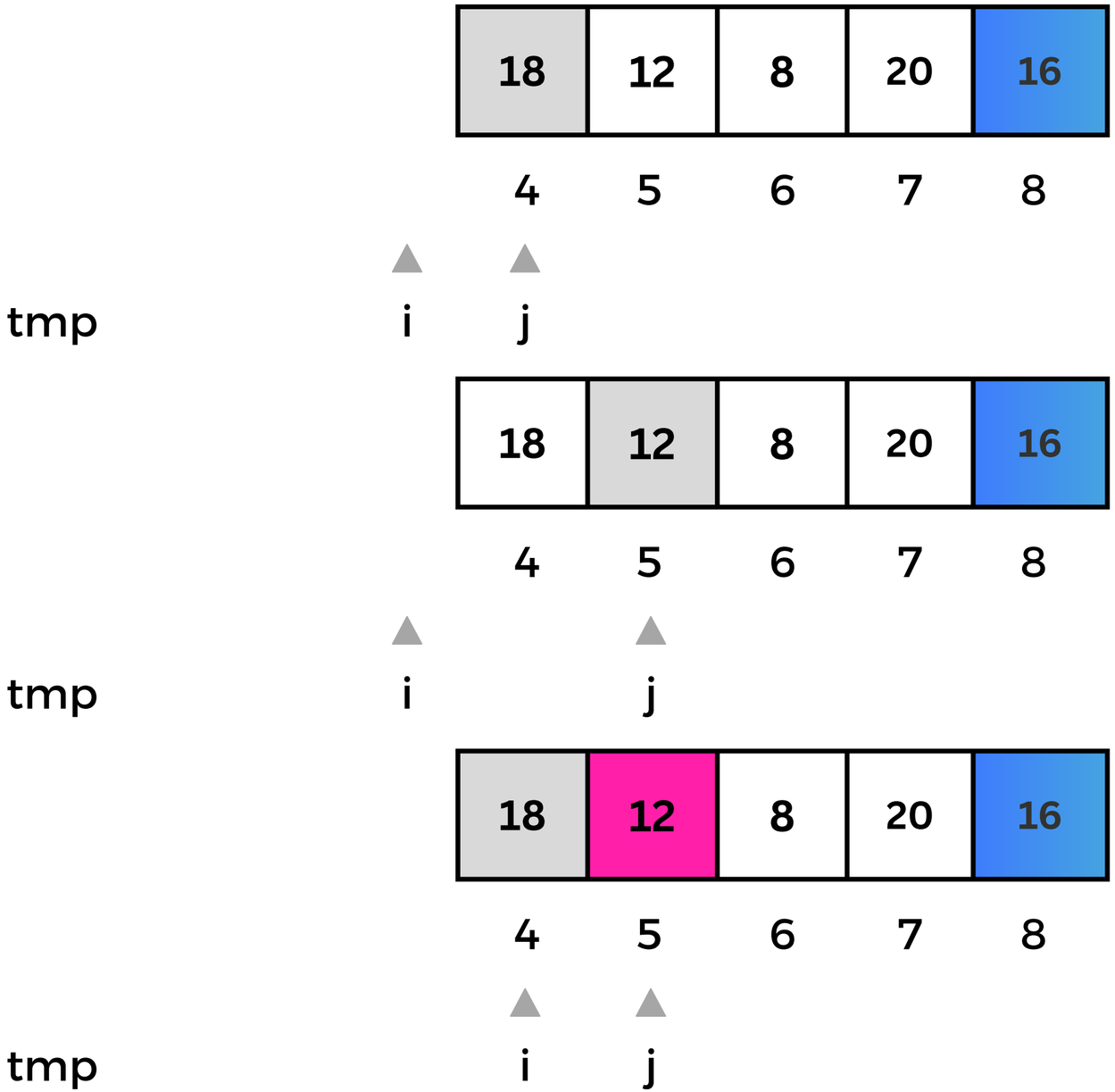


✤ PRINCIPE DE FONCTIONNEMENT.

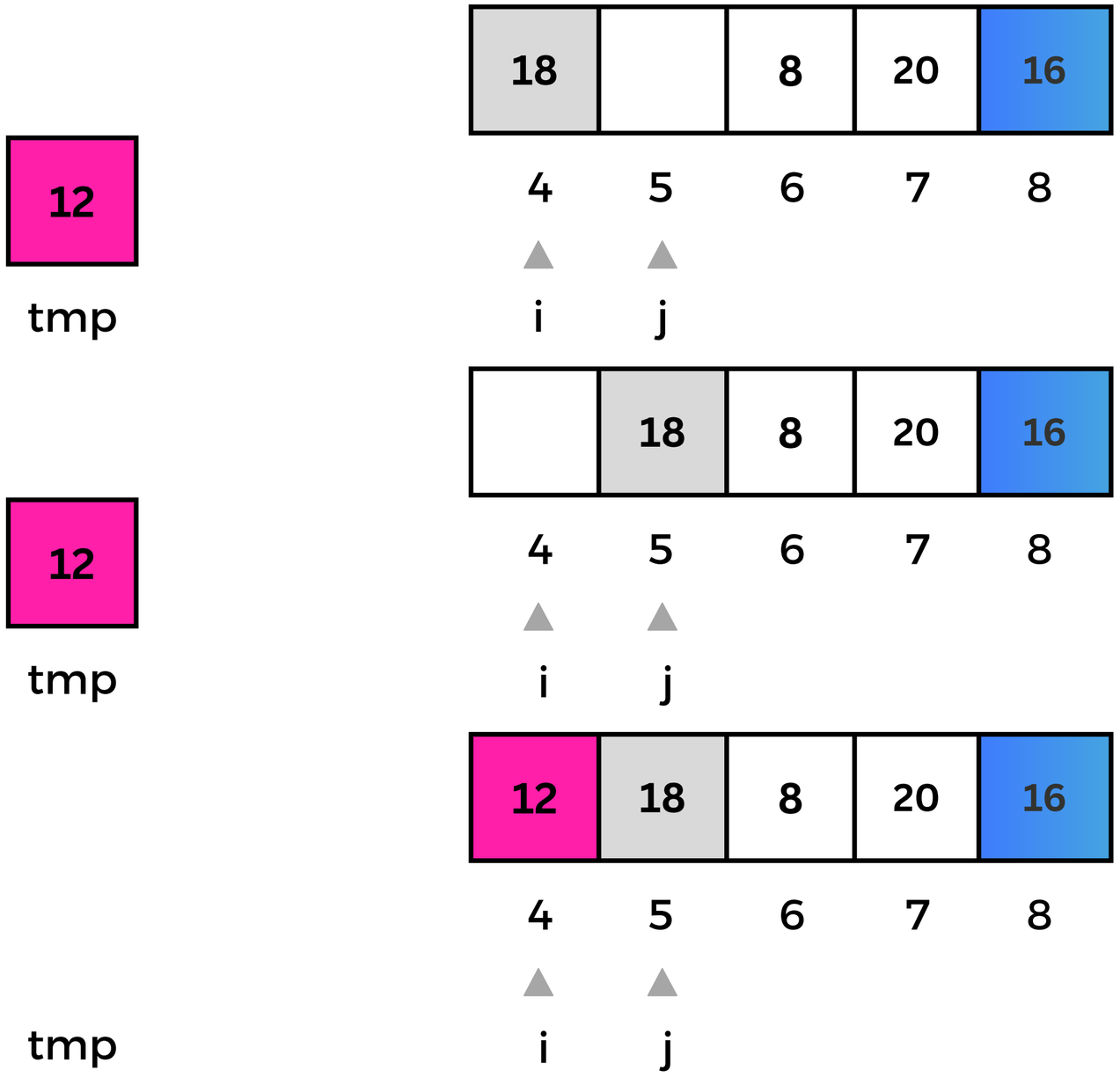
🔧 Étape 2.2 : Tri Ls= [18,12,8,20,16]



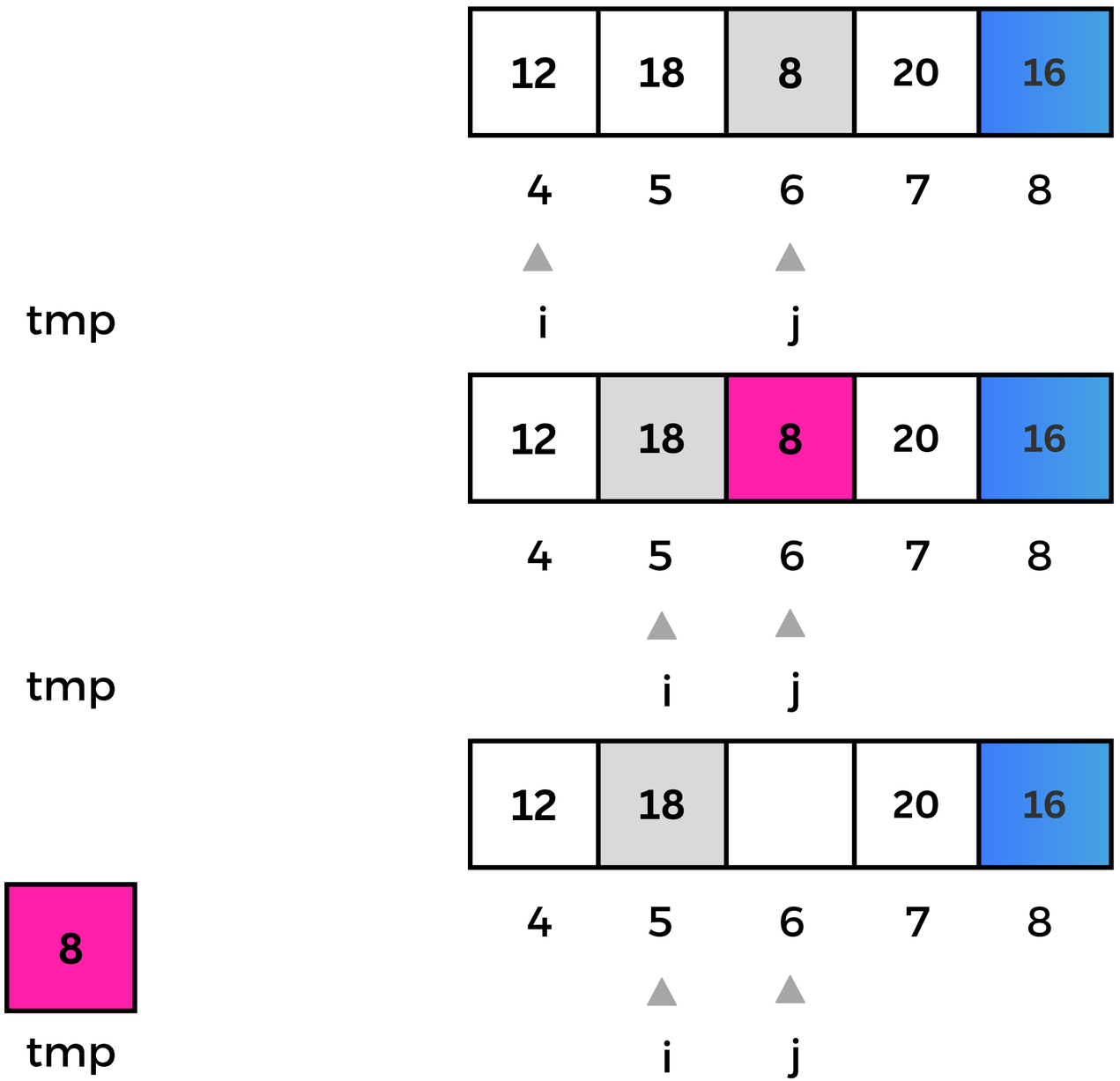
✤ PRINCIPE DE FONCTIONNEMENT.



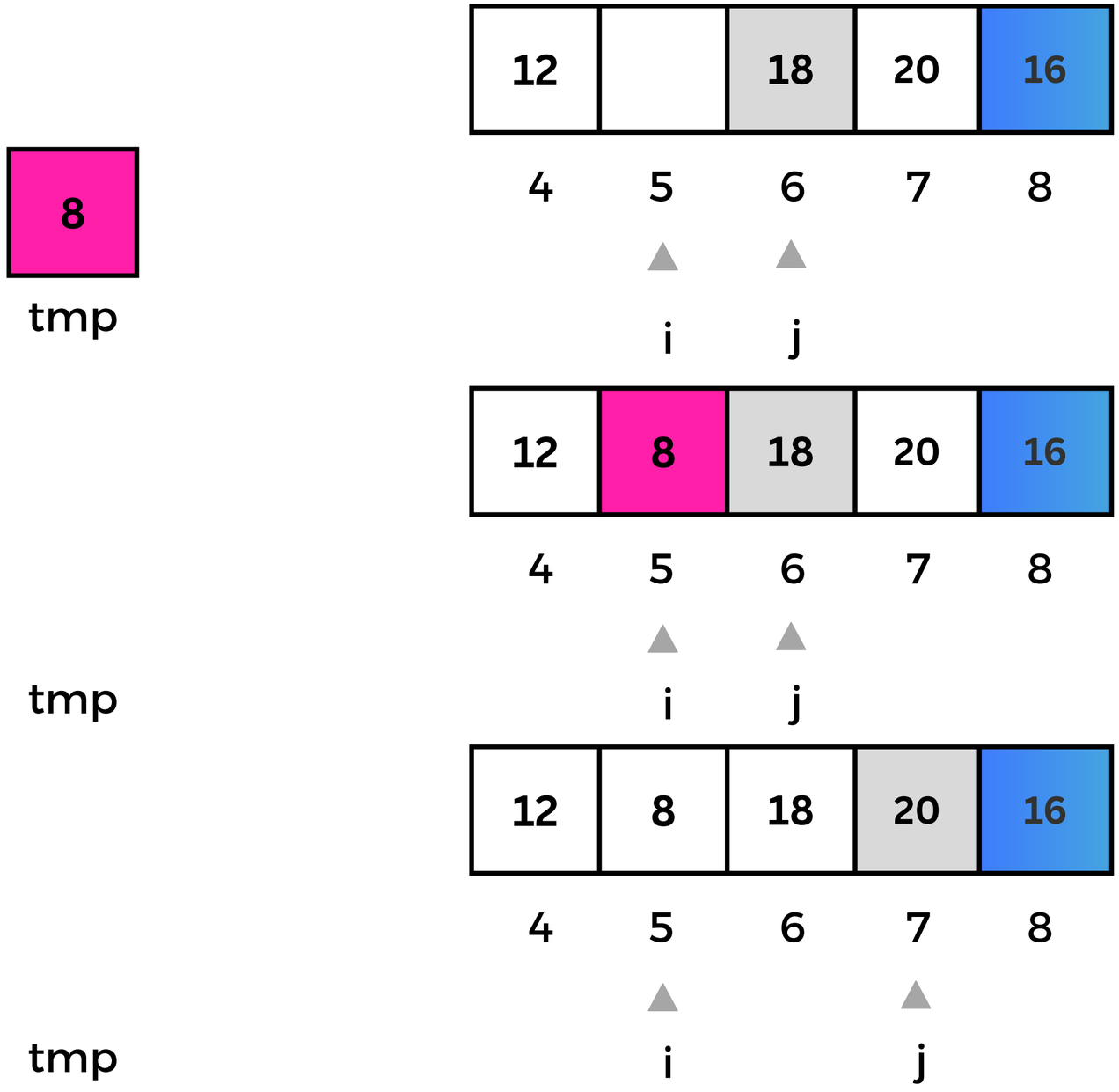
✤ PRINCIPE DE FONCTIONNEMENT.



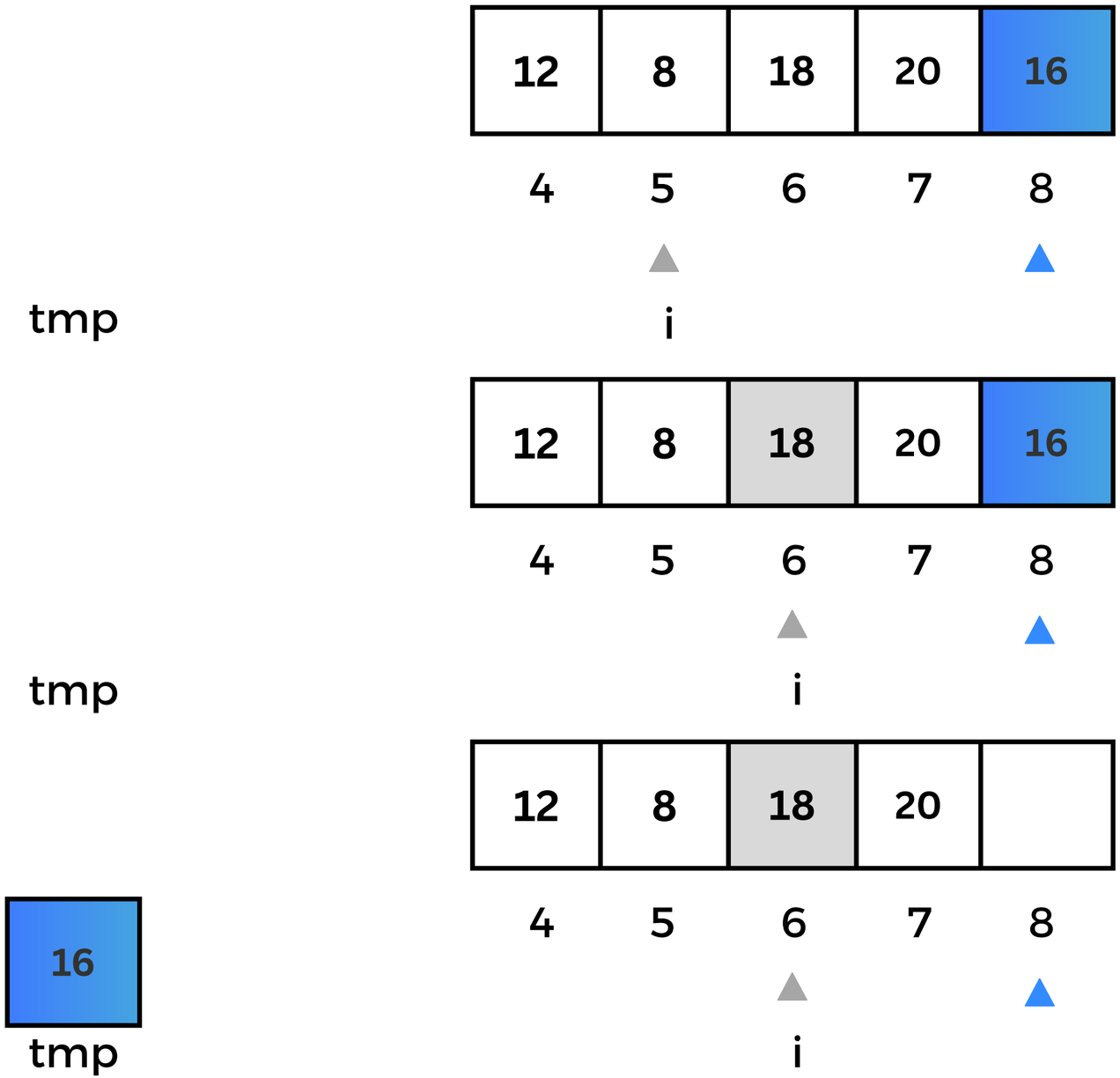
✤ PRINCIPE DE FONCTIONNEMENT.



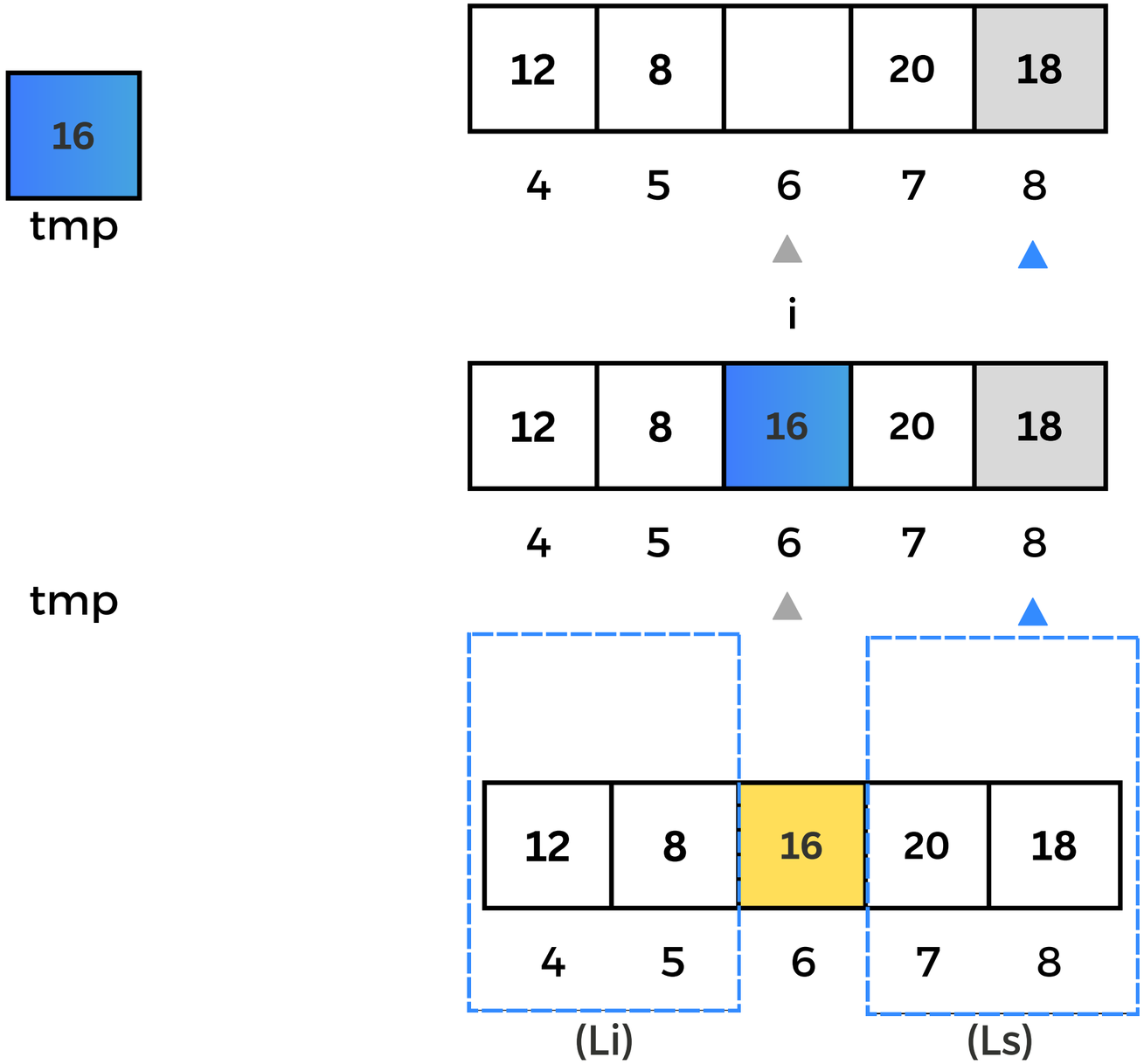
✤ PRINCIPE DE FONCTIONNEMENT.



✤ PRINCIPE DE FONCTIONNEMENT.

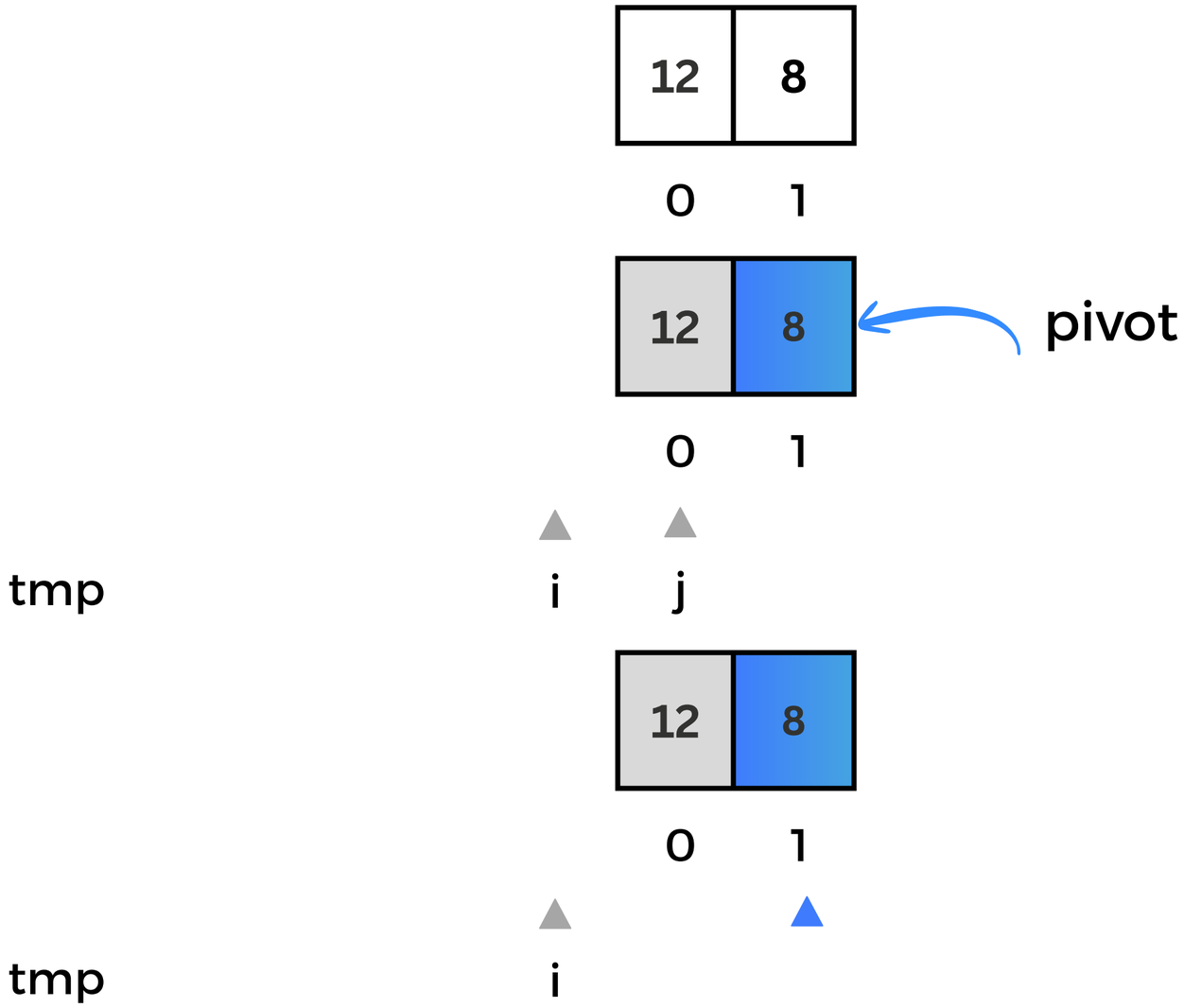


✤ PRINCIPE DE FONCTIONNEMENT.

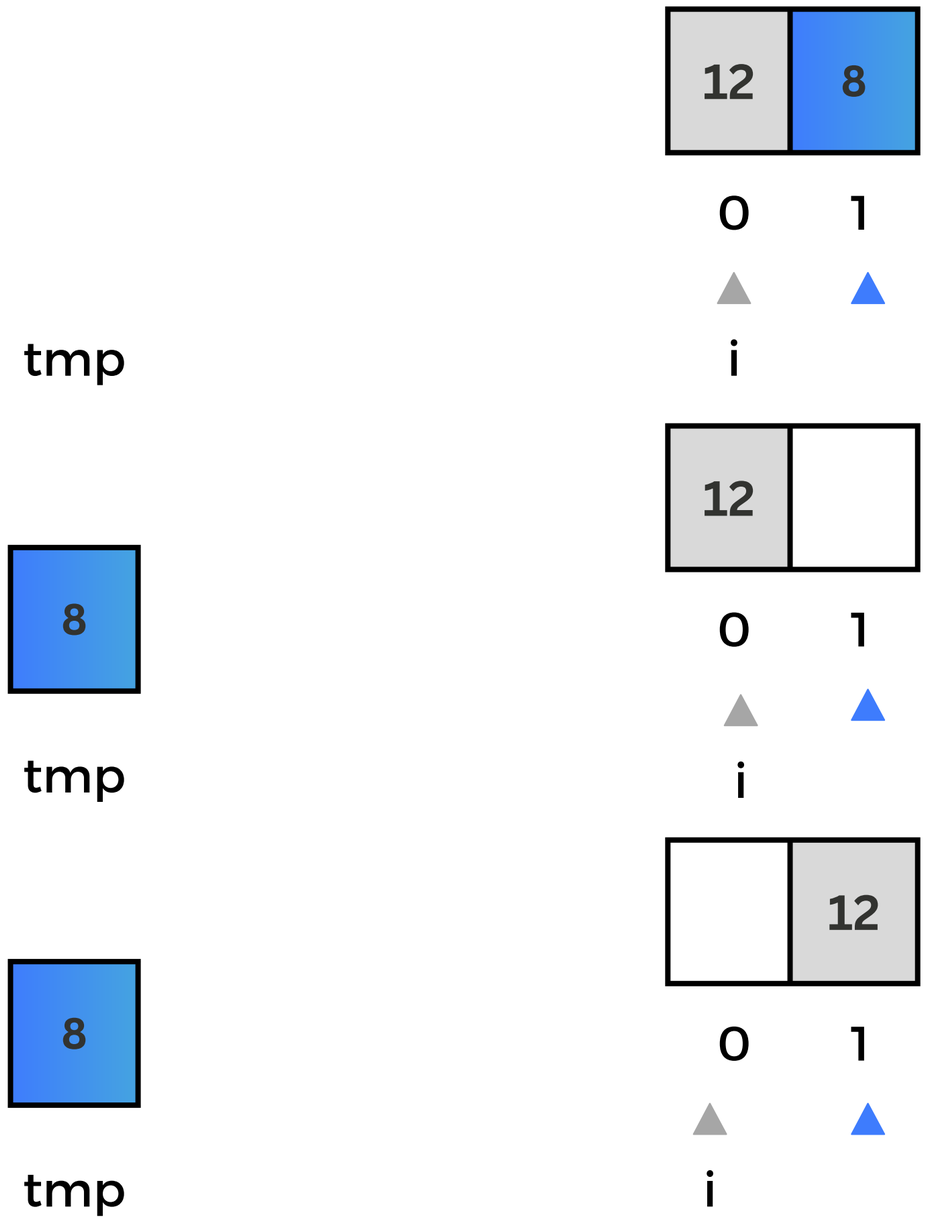


✤ PRINCIPE DE FONCTIONNEMENT.

🔧 Étape 2.2.1 : Tri Li= [12,8]



✤ PRINCIPE DE FONCTIONNEMENT.

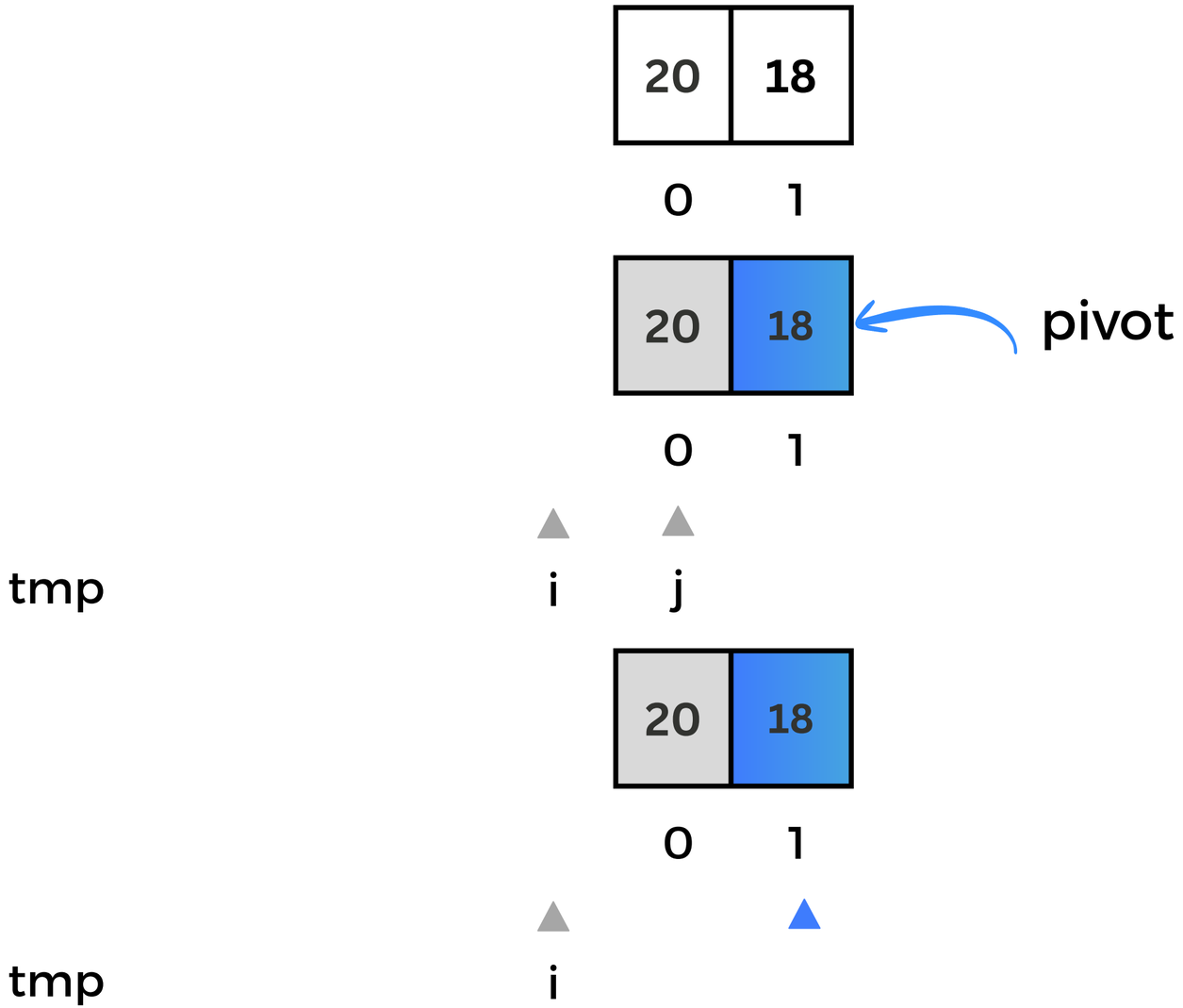


✤ PRINCIPE DE FONCTIONNEMENT.

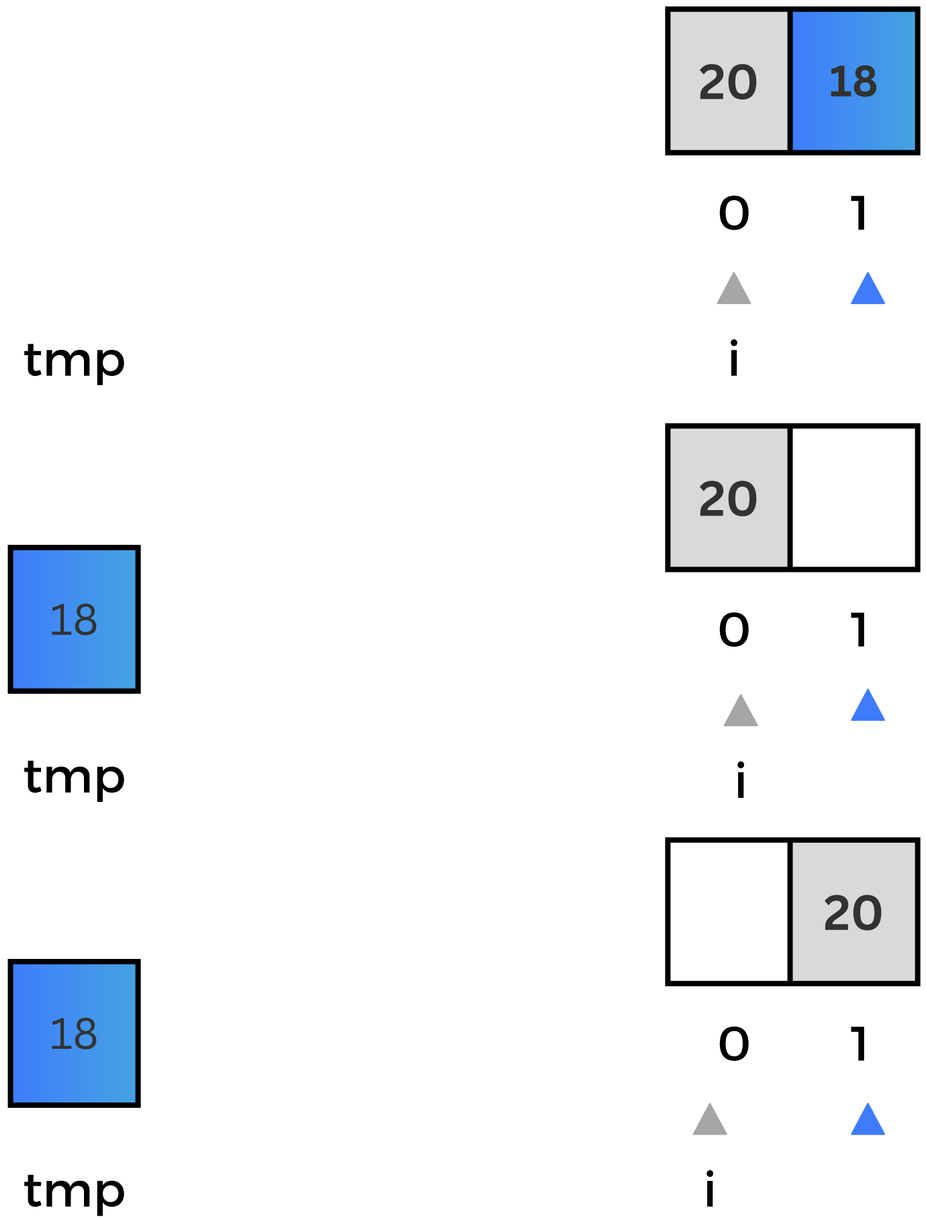
📊 **Résultat de l’étape 2.2.1: :**

✤ PRINCIPE DE FONCTIONNEMENT.

🔧 Étape 2.2.2 : Tri Li= [20,18]

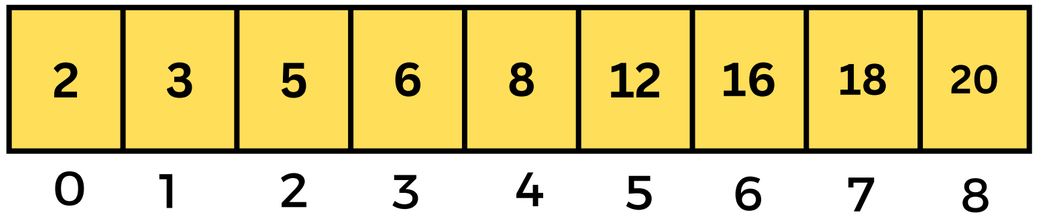


✤ PRINCIPE DE FONCTIONNEMENT.

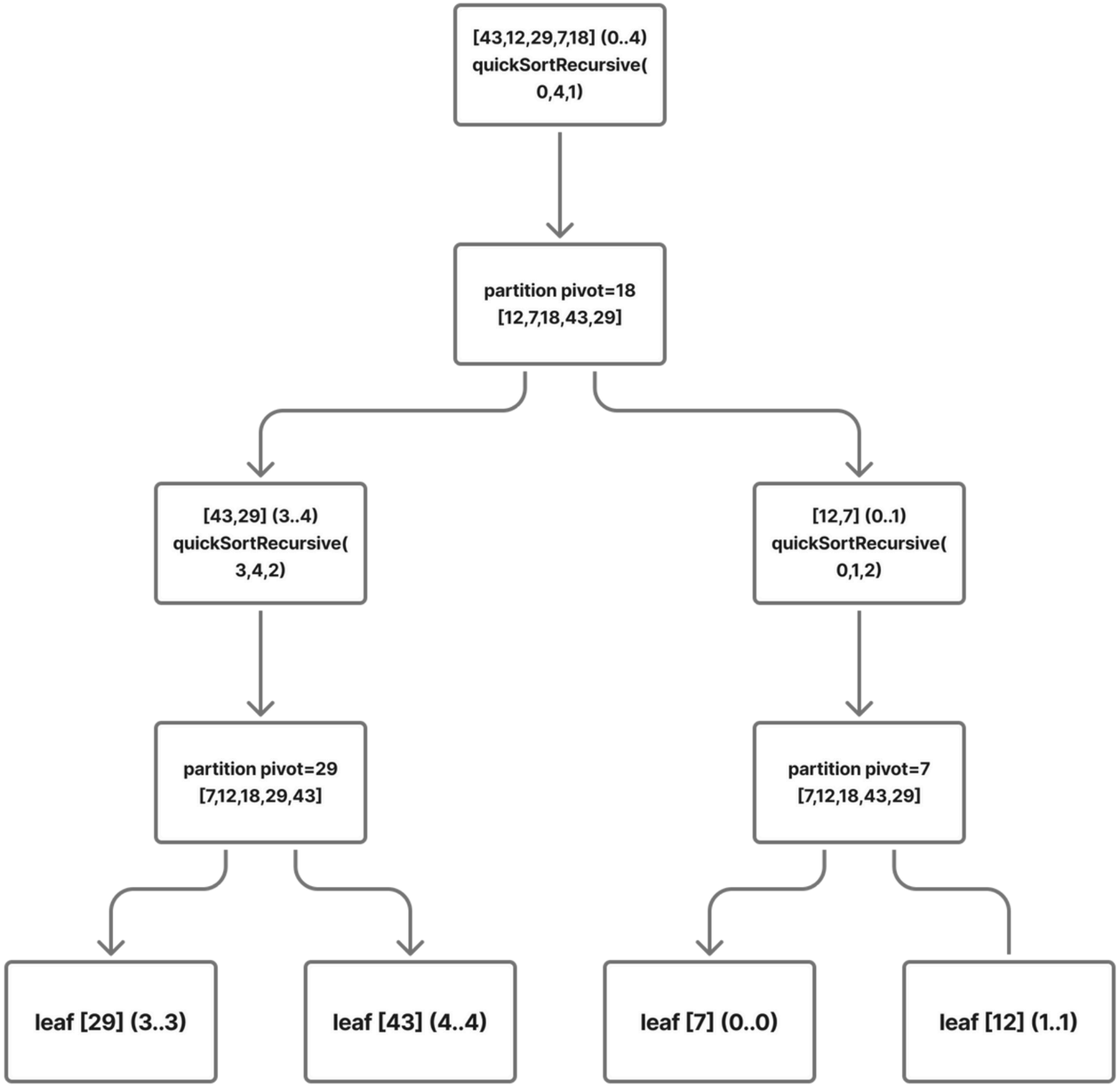


✤ PRINCIPE DE FONCTIONNEMENT.

✅ **Liste triée avec Tri Rapide:**



✤ IMPLEMENTATION



✤ IMPLEMENTATION

👉 algorithme de Tri Rapide:

```
Algorithme TriRapide( G  , D : entier );
Local :  i : entier
début
 si D > G alors
    i ¬ Partition( G , D );
   TriRapide( G , i-1 );
   TriRapide( i+1 , D );
 Fsi
FinTRiRapide
```

✤ IMPLEMENTATION

👉 la fonction partition:

```
fonction Partition( G , D : entier )  résultat : entier
Local : i , j ,  piv , temp : entier
début
    piv  ¬  Tab[D];
    i  ¬ G-1;
    j  ¬ D;
    repeter
      repeter  i ¬  i+1 jusquà Tab[i] >= piv;
      repeter  j ¬  j-1 jusquà Tab[j] <= piv;
      temp ¬  Tab[i];
      Tab[i] ¬  Tab[j];
      Tab[j] ¬  temp
    jusquà j <= i;
    Tab[j] ¬  Tab[i];
    Tab[i] ¬  Tab[d];
    Tab[d] ¬  temp;
    résultat ¬  i
FinPartition
```

✤ COMPLEXITÉ.

👉 COMPLEXITÉ DANS LE MEILLEUR CAS:

La formule de récursion est :

$$T(n)=2T(n/2)+n$$

👉 Le **n** vient du coût du partitionnement (on parcourt une fois le tableau).

🟦 Démonstration avec le Master Théorème:

forme Générale:

$$T(n)=aT(n/b)+C \qquad \text{telle que: } a=2 , b=2 , C=n$$

On calcule :

$$\log b(a)=\log 2(2)=1$$

Comme :

$$f(n)=n=n^{(\log b(a))}$$

👉 On est dans le cas 2 du Master Theorem. **$T(n)=\Theta(n \log n)$**

✤ COMPLEXITÉ.

👉 COMPLEXITÉ DANS LE PIRE CAS:

Dans ce cas, il n’y a plus de division, seulement une réduction de 1 :

$$T(n)=T(n-1)+1$$

Développons la récursion :

$$T(n)=(n)+(n-1)+(n-2)+\ldots+1$$

C’est la somme :

$$T(n)=n(n-1)/2$$

Donc :

$$T(n)=\Theta(n^2)$$

Merci Pour Votre Attention