I21: Introduction à l'algorithmique

Cours 4: Algorithmes de tris

Nicolas Méloni

Licence 1 (2017-2020)

Problème du tri

Problème : Tri

Entrée : tableau d'entiers T de taille n
Sortie : une permutation des éléments de T telle que

$$T[1] \leq T[2] \leq \cdots \leq T[n].$$

Problème du tri

- Trier des données est la base de nombreux algorithmes :
  - recherche d'élément
  - plus proche paire
  - unicité
  - enveloppe convexe
- Les algorithmes de tris font appels à des idées réutilisables dans de nombreux autres contextes.
- C'est historiquement le problème le plus étudié en informatique.

Les tris standards

Trois algorithmes *classiques* :

- tri par sélection
- tri par propagation
- tri par insertion

Caractéristiques des tris standards

Ces tris partagent 2 grandes caractéristiques :

Tris Comparatifs

Ils reposent exlusivement sur l'opération de comparaison des éléments.

Tris *in situ*

Seul un nombre constant d'éléments est stocké hors du tableau à trié.

Caractéristiques des tris standards

L'opération principale est l'échange de deux valeurs du tableau.

```
1  ALGORITHME Swap(T,i,j):
2  DONNEES
3     T: tableau d entiers
4     i,j: entiers
5  VARIABLES:
6     aux: entier
7  DEBUT
8     aux ← T[i]
9     T[i] ← T[j]
10    T[j] ← aux
11 FIN
```

- complexité : $O(1)$

Tri par sélection

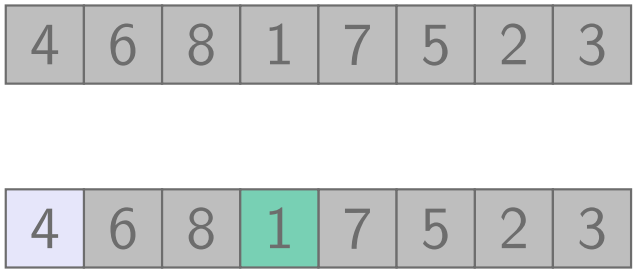
Idée générale :

- Chercher le plus petit élément du tableau ;
- l'échanger avec l'élément d'indice 1 ;
- chercher le deuxième plus petit élément ;
- l'échanger avec l'élément d'indice 2 ;
- etc

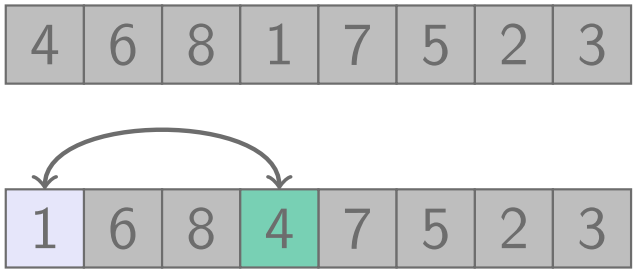
Tri par sélection

4	6	8	1	7	5	2	3

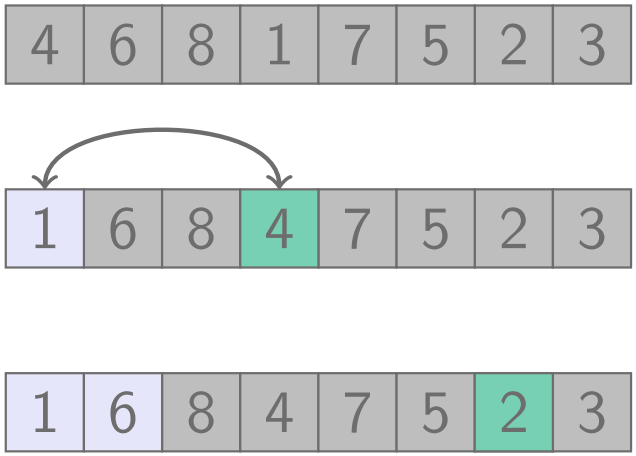
Tri par sélection



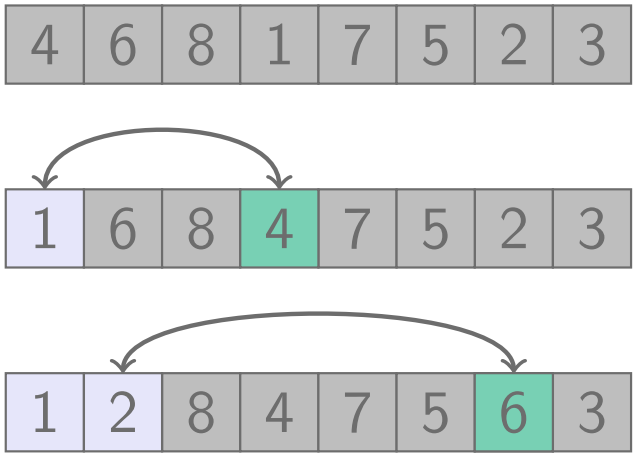
Tri par sélection



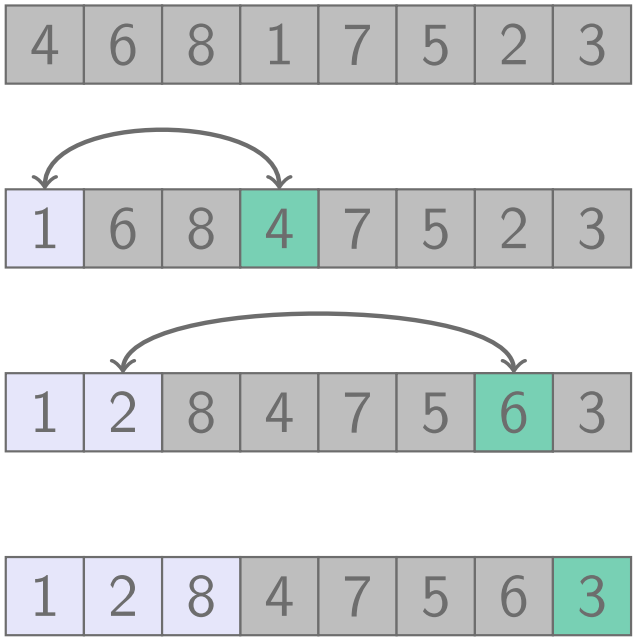
Tri par sélection



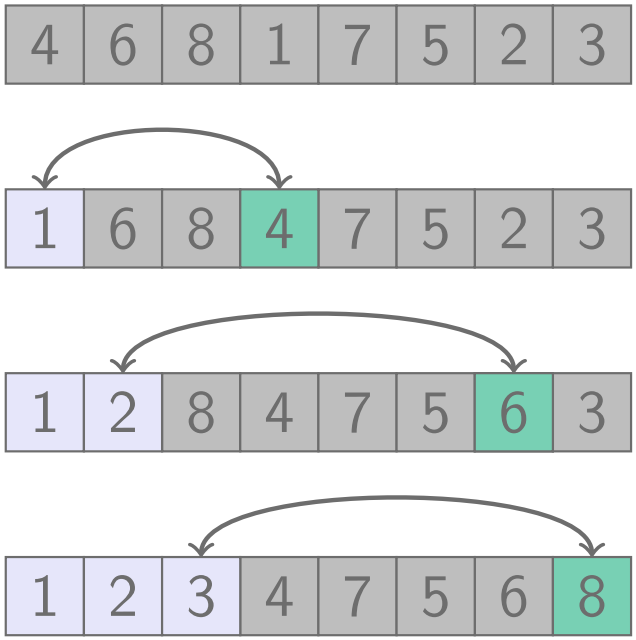
Tri par sélection



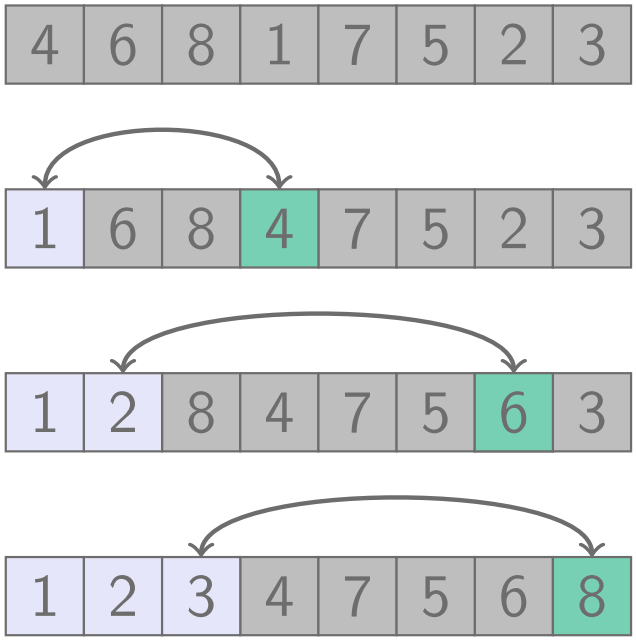
Tri par sélection



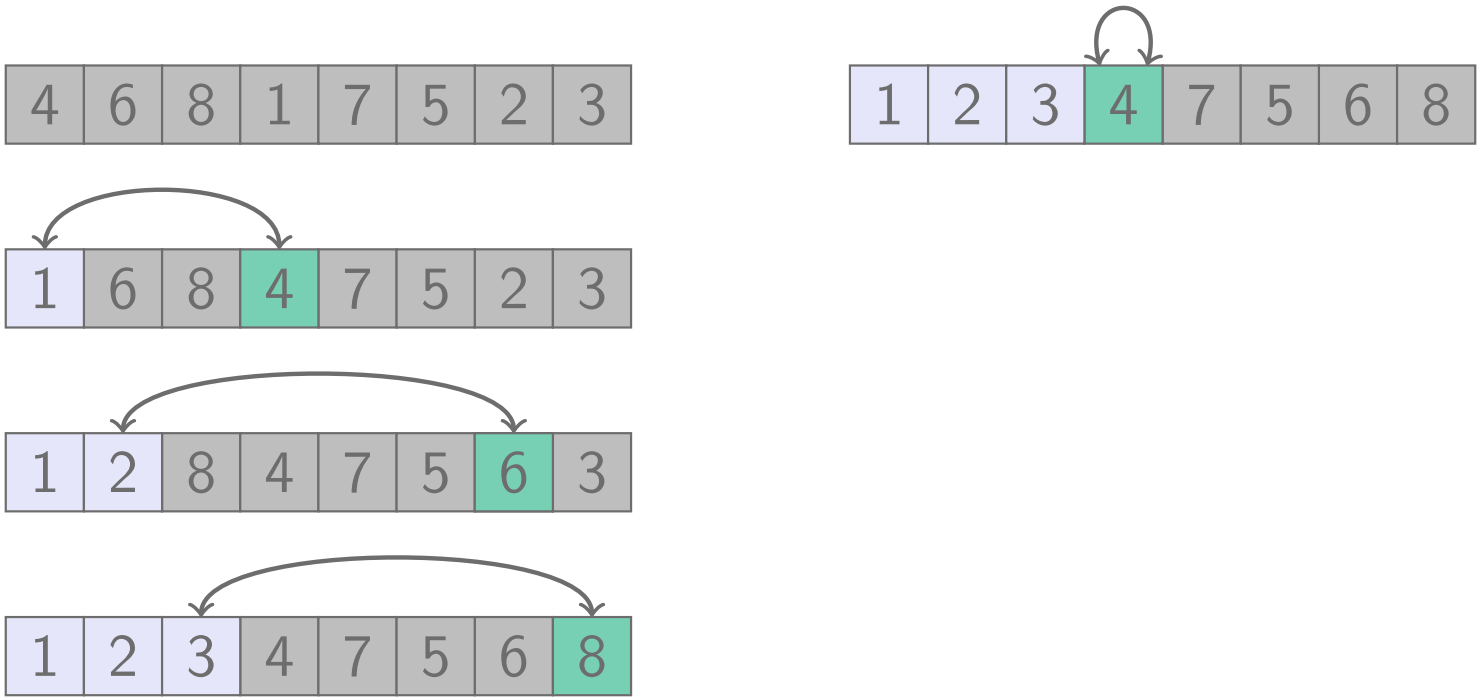
Tri par sélection



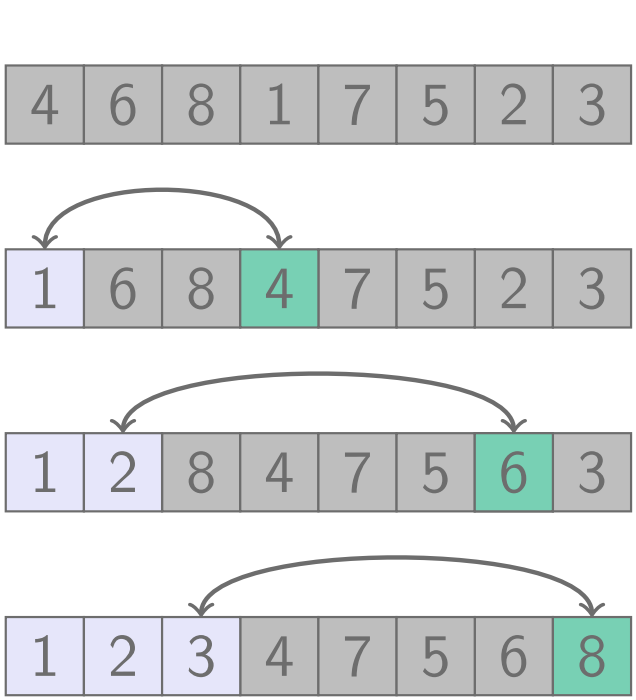
Tri par sélection

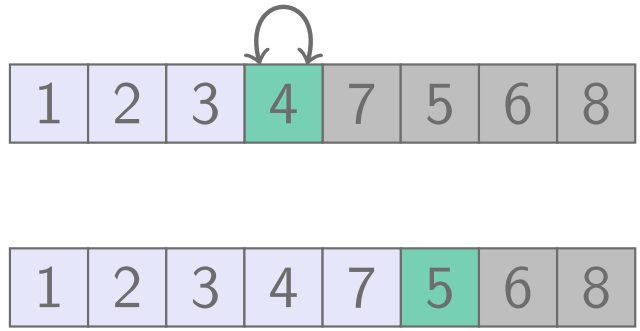


Tri par sélection

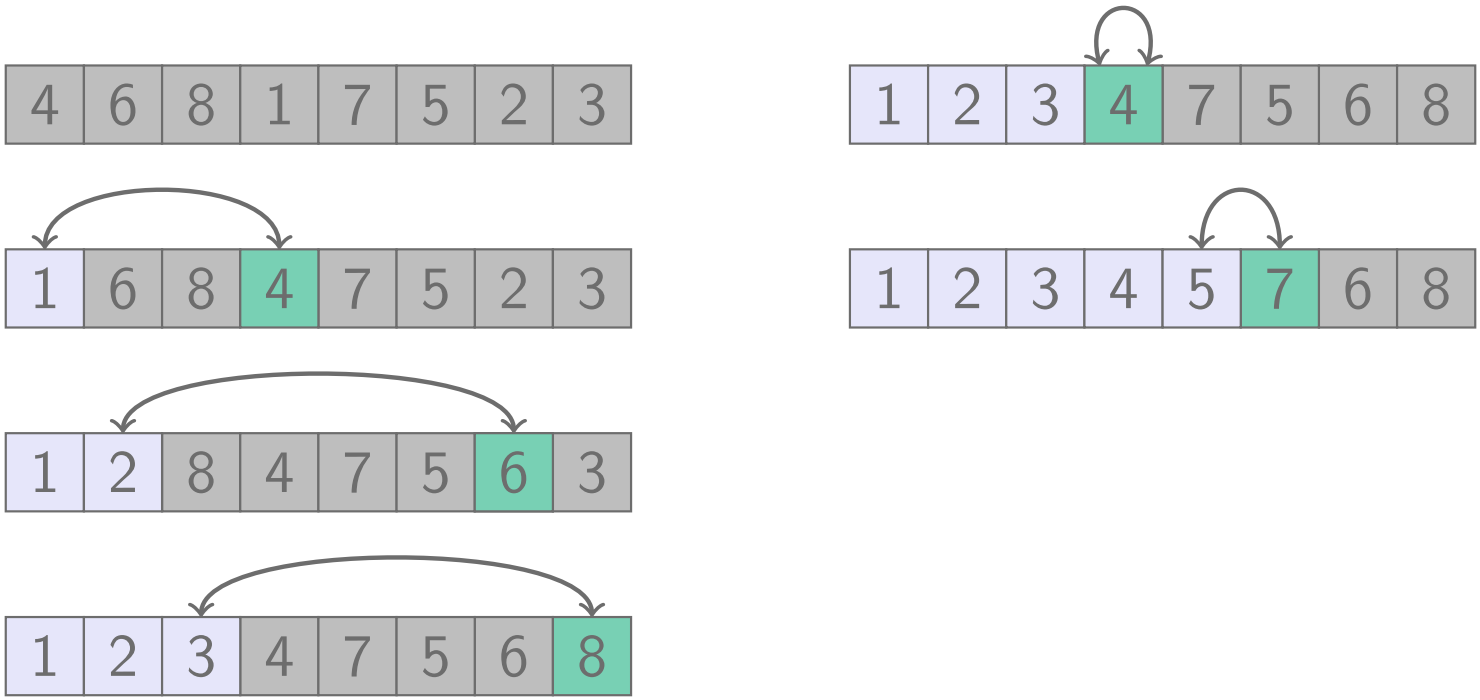


Tri par sélection

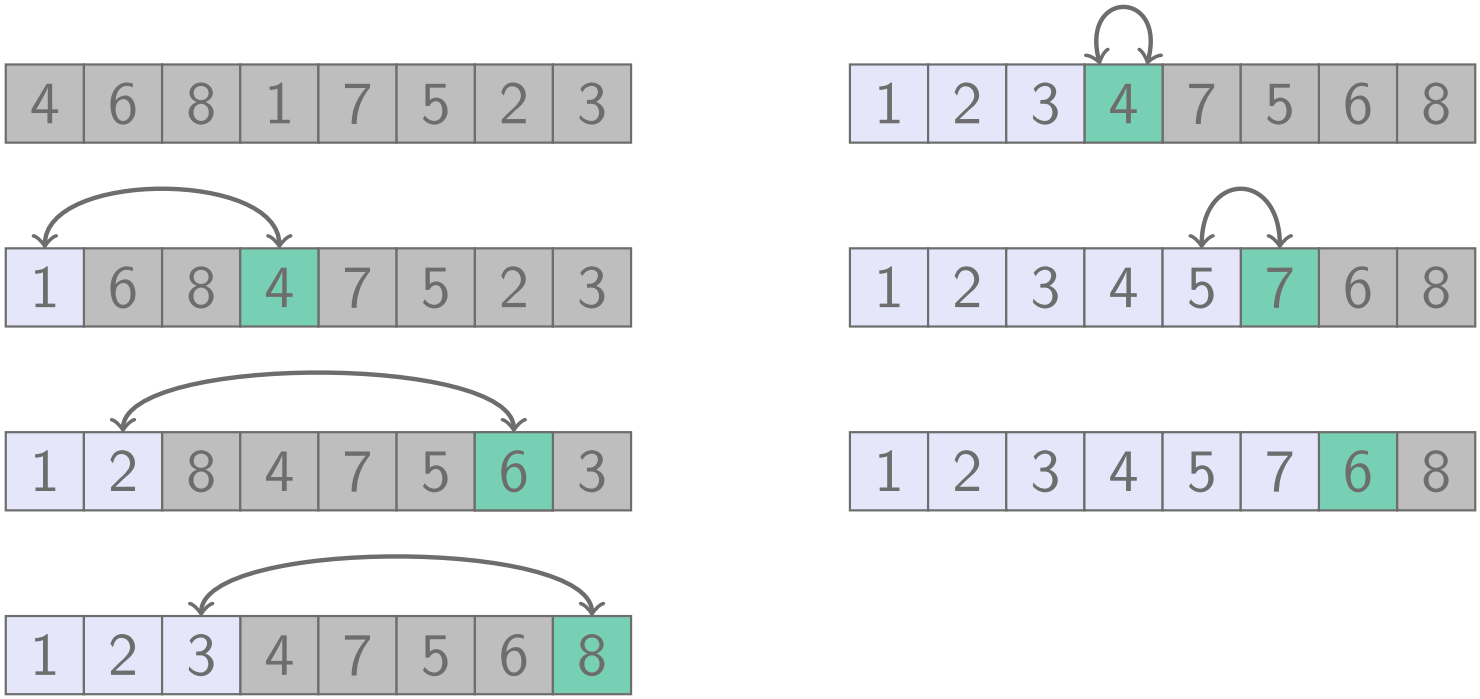




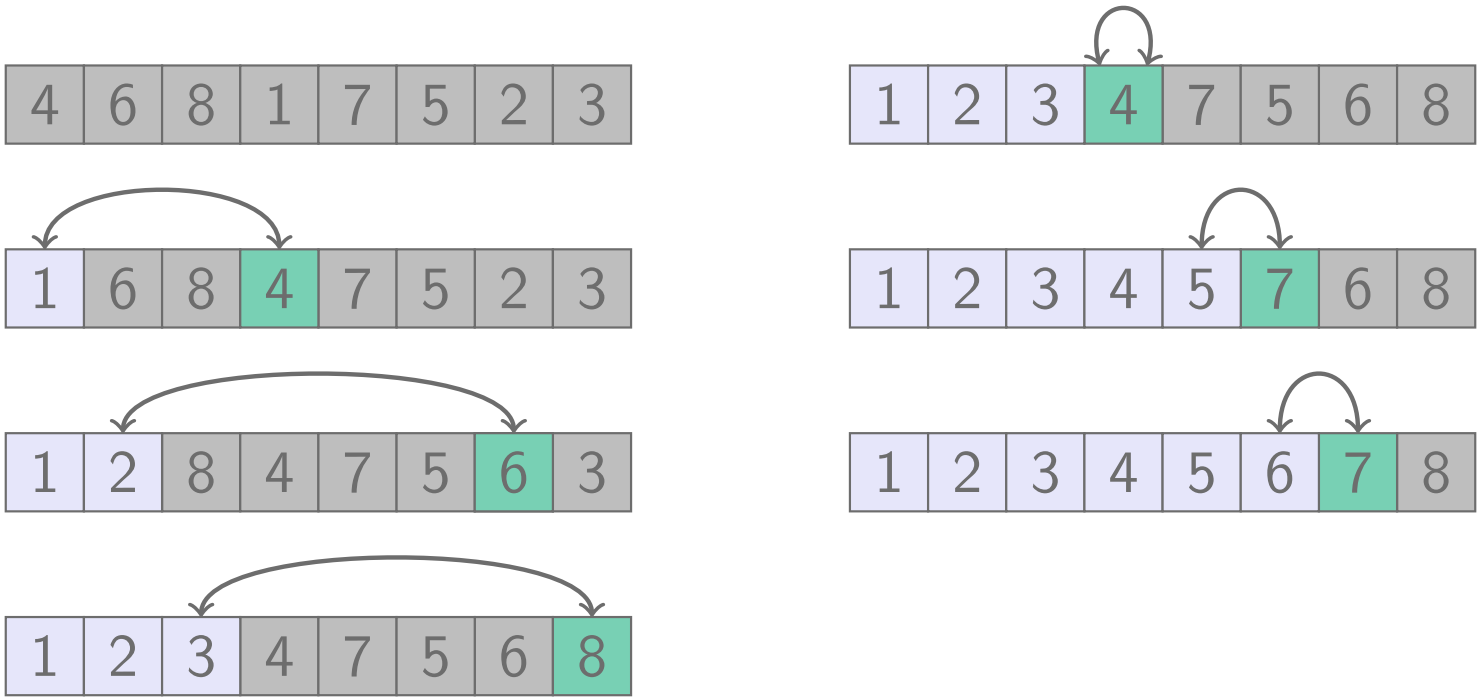
Tri par sélection



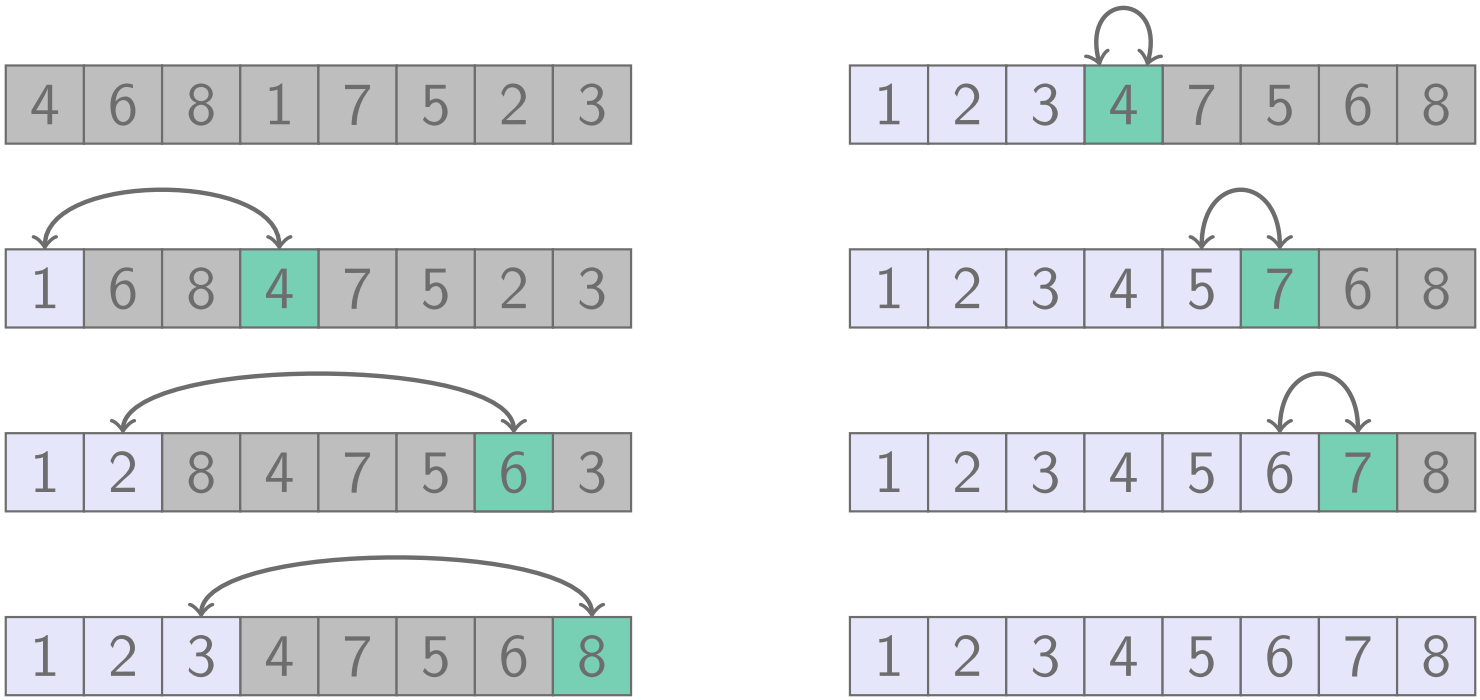
Tri par sélection



Tri par sélection



Tri par sélection



Tri par sélection

```
1  ALGORITHME TriSelection(T):
2  DONNEES
3    T: tableau d'entiers de taille n
4  VARIABLES
5    imin,i,j: entiers
6  DEBUT
7    i←1
8    TQ i ≤ n−1 FAIRE
9      imin←i
10     j←i+1
11     TQ j ≤ n FAIRE
12       SI T[j] < T[imin] ALORS
13         imin←j
14       FSI
15       j←j+1
16     FTQ
17     Swap(T,i,imin)
18     i←i+1
19   FTQ
20 FIN
```

Tri par sélection

```
1  ALGORITHME TriSelection(T):
2  DONNEES
3    T: tableau d'entiers de taille n
4  VARIABLES
5    imin,i,j: entiers
6  DEBUT
7    i←1
8    TQ i ≤ n−1 FAIRE
9      imin←i
10     j←i+1
11     TQ j ≤ n FAIRE
12       SI T[j] < T[imin] ALORS
13         imin←j
14       FSI
15       j←j+1
16     FTQ
17     Swap(T,i,imin)
18     i←i+1
19   FTQ
20 FIN
```

- Arrêt : deux boucles imbriquées à incrément constant
- Validité : $(T[1:i-1]$ est trié et $T[i-1] \leq \min(T[i:n]))$ est un invariant de boucle
- Complexité : $\Theta(n^2)$

Tri par propagation (ou tri à bulles)

Idée générale :

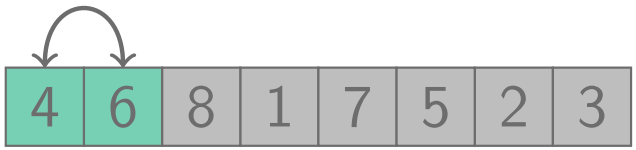
- Faire remonter les élément les plus grands en échangeant les éléments contigu mal arrangés ;
- effectuer autant de passes que nécessaire pour que tous les éléments soient à la bonne place.

Tri par propagation (ou tri à bulles)

6	4	8	1	7	5	2	3

Tri par propagation (ou tri à bulles)

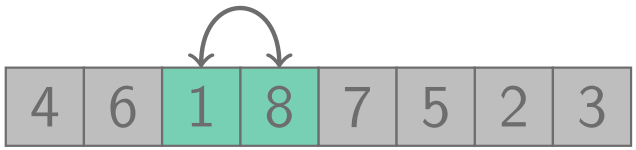
Tri par propagation (ou tri à bulles)



Tri par propagation (ou tri à bulles)

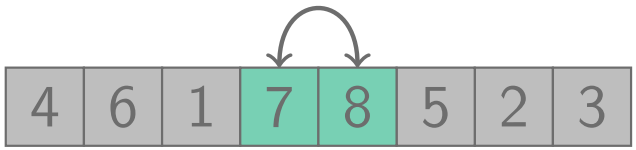
Tri par propagation (ou tri à bulles)

Tri par propagation (ou tri à bulles)



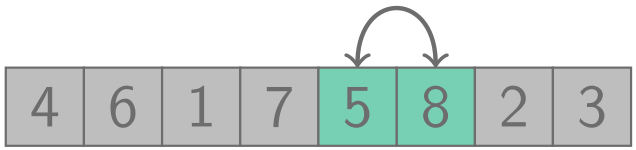
Tri par propagation (ou tri à bulles)

Tri par propagation (ou tri à bulles)



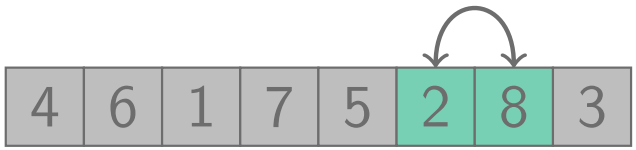
Tri par propagation (ou tri à bulles)

Tri par propagation (ou tri à bulles)



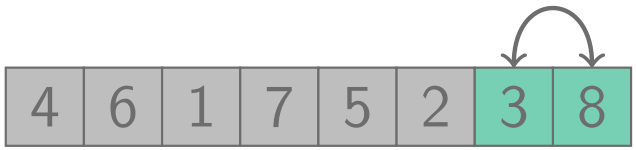
Tri par propagation (ou tri à bulles)

Tri par propagation (ou tri à bulles)

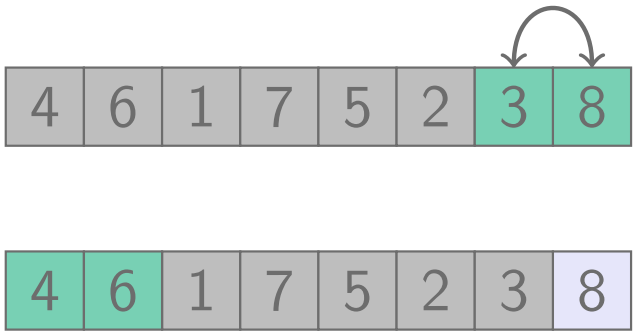


Tri par propagation (ou tri à bulles)

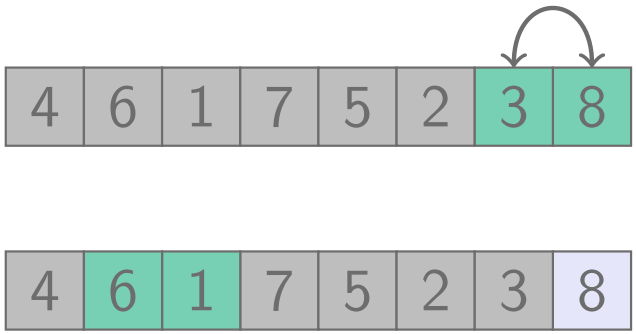
Tri par propagation (ou tri à bulles)



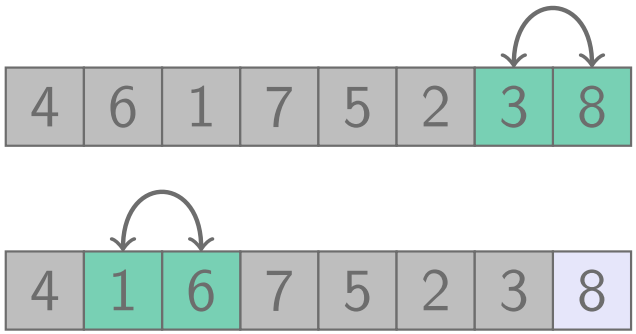
Tri par propagation (ou tri à bulles)



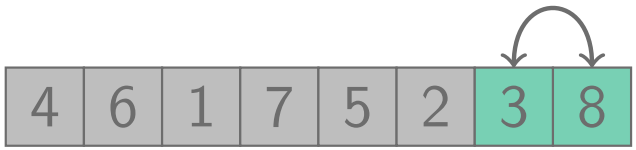
Tri par propagation (ou tri à bulles)



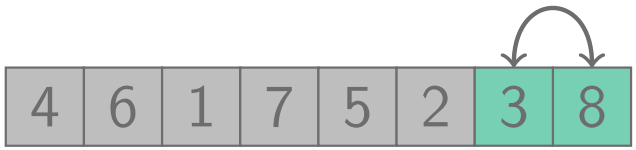
Tri par propagation (ou tri à bulles)



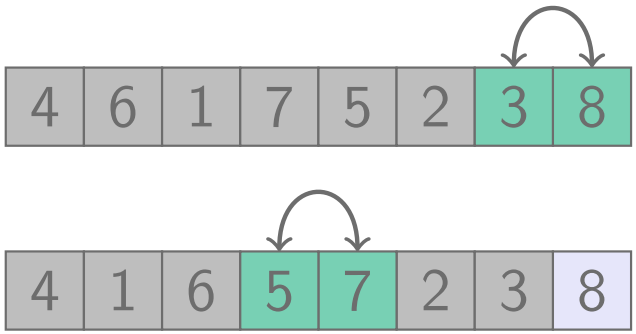
Tri par propagation (ou tri à bulles)



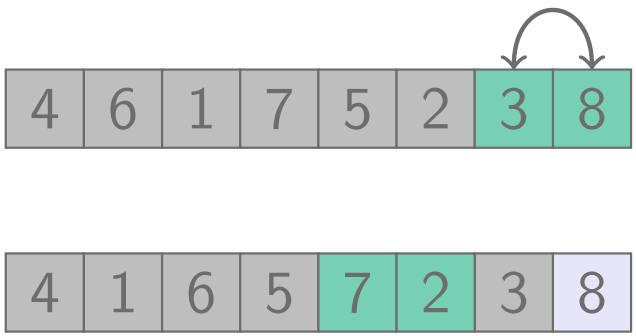
Tri par propagation (ou tri à bulles)



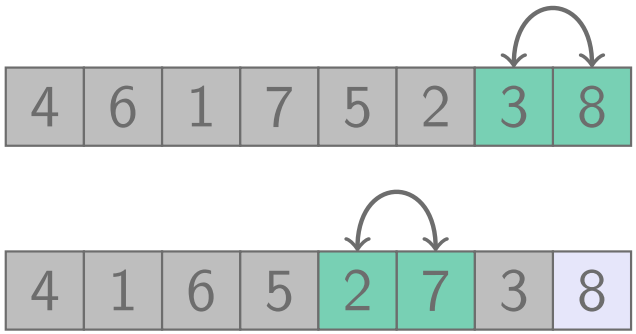
Tri par propagation (ou tri à bulles)



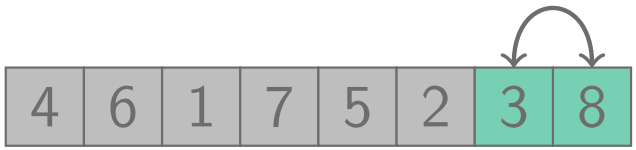
Tri par propagation (ou tri à bulles)



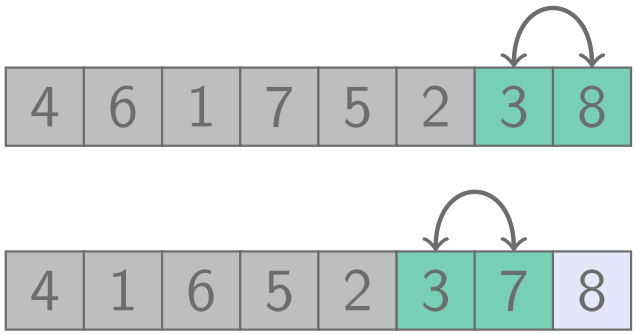
Tri par propagation (ou tri à bulles)



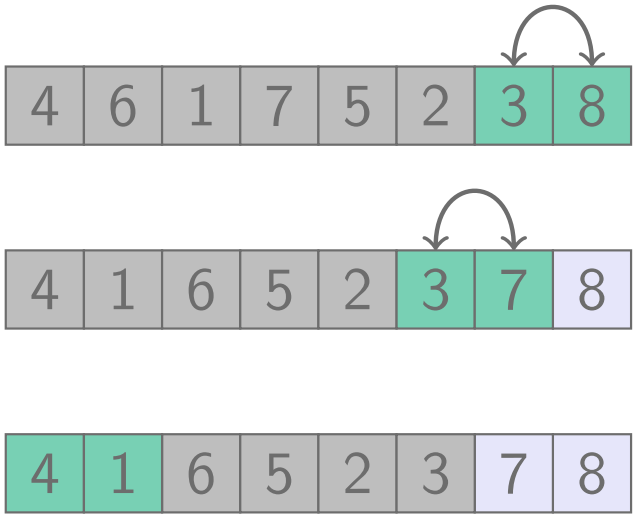
Tri par propagation (ou tri à bulles)



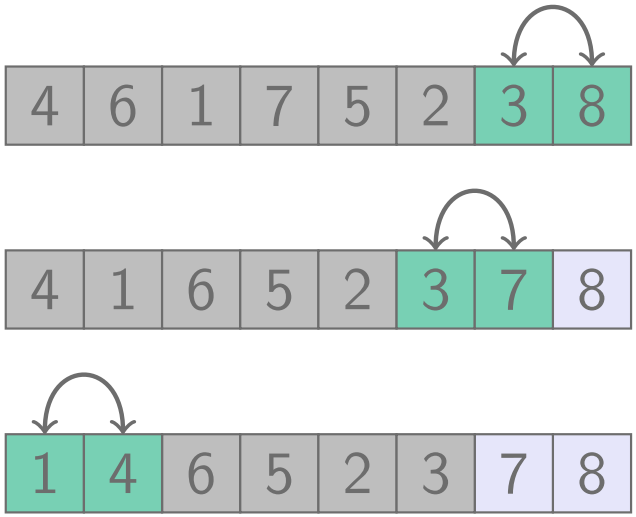
Tri par propagation (ou tri à bulles)



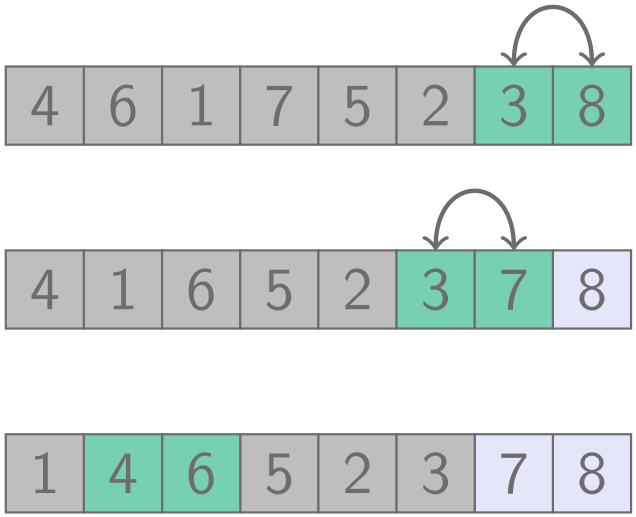
Tri par propagation (ou tri à bulles)



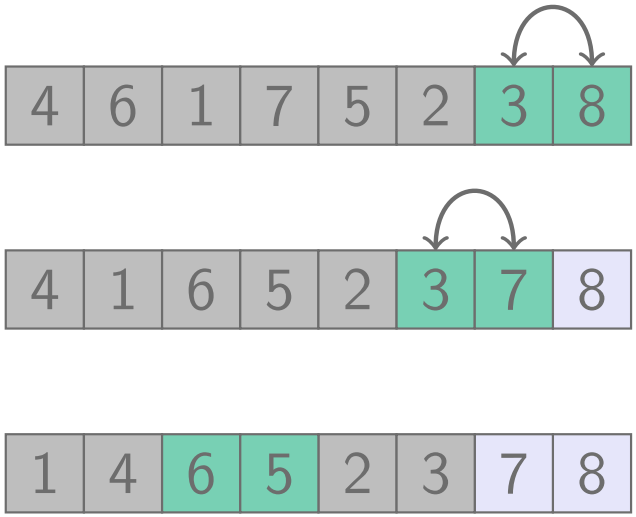
Tri par propagation (ou tri à bulles)



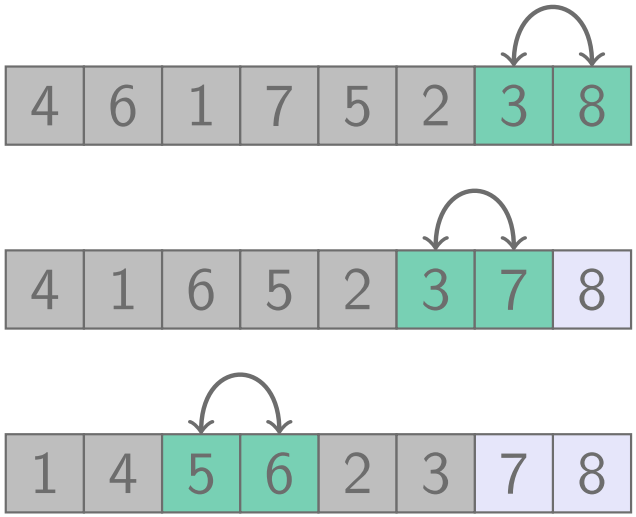
Tri par propagation (ou tri à bulles)



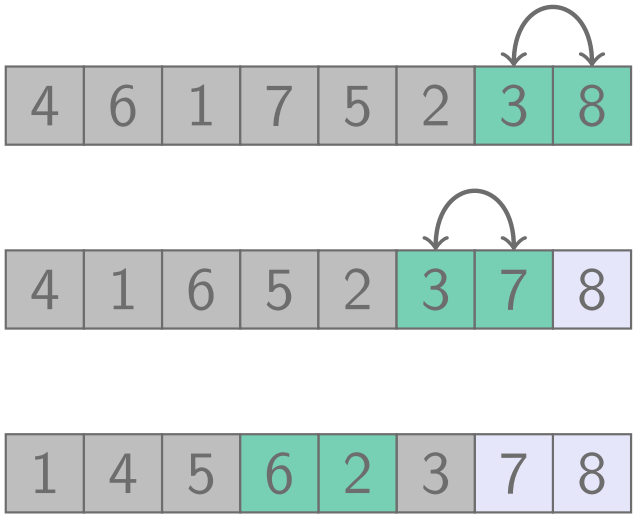
Tri par propagation (ou tri à bulles)



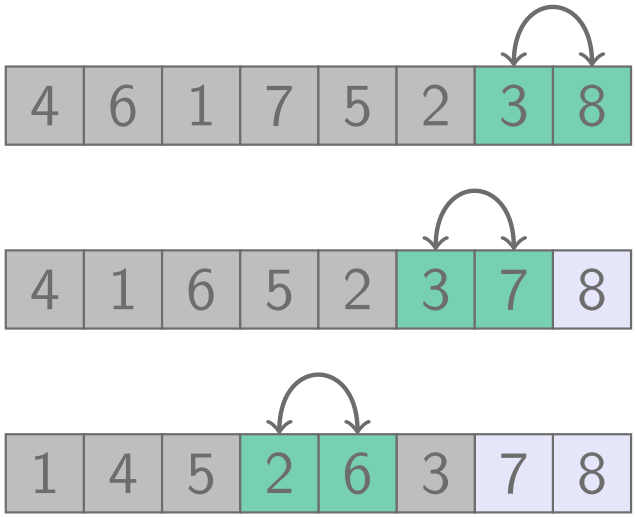
Tri par propagation (ou tri à bulles)



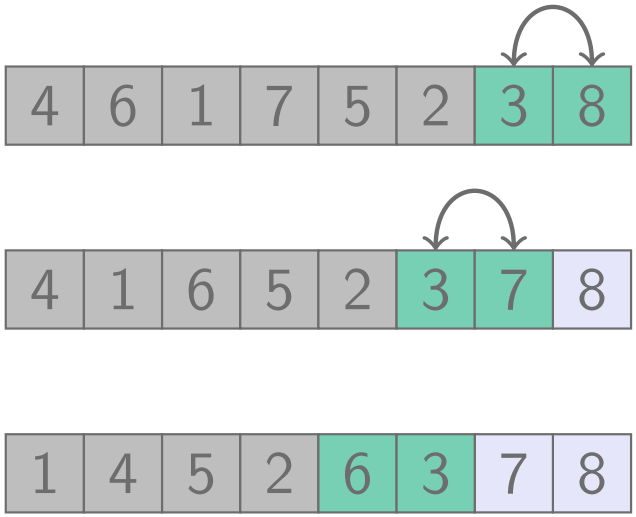
Tri par propagation (ou tri à bulles)



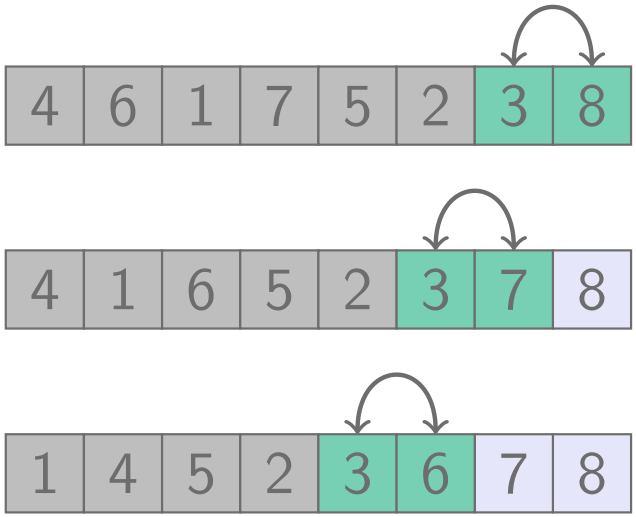
Tri par propagation (ou tri à bulles)



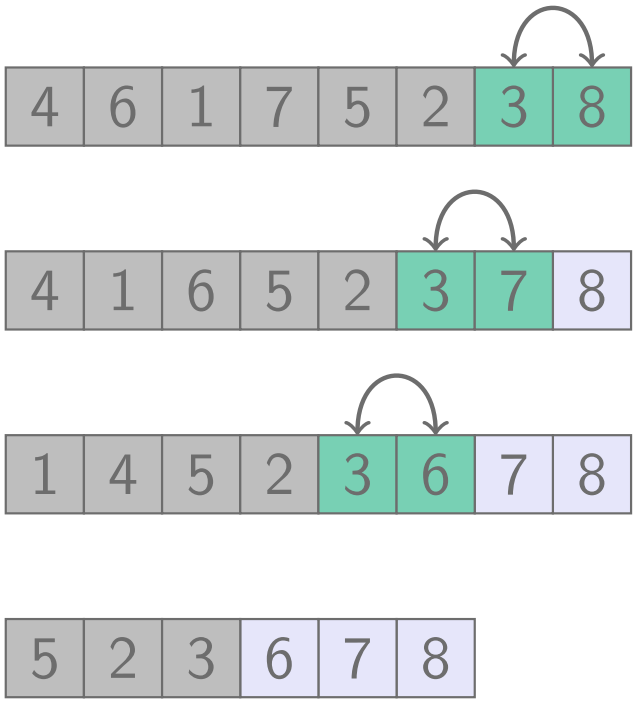
Tri par propagation (ou tri à bulles)



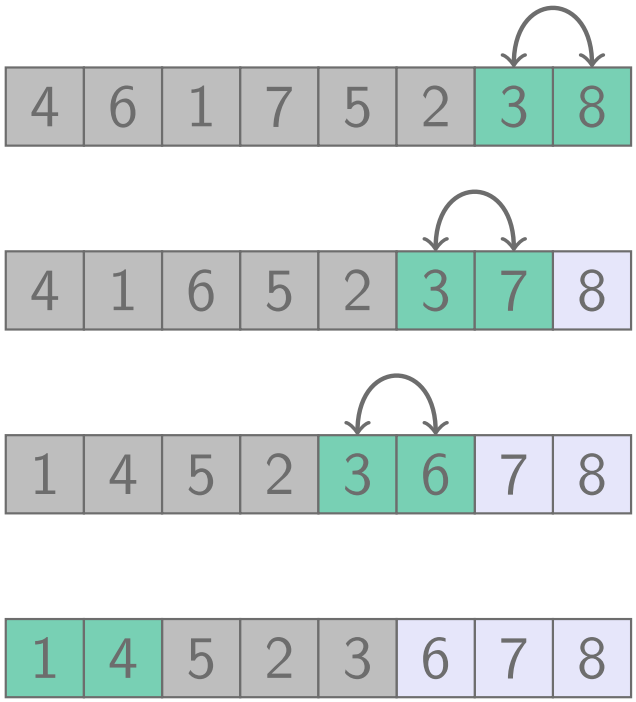
Tri par propagation (ou tri à bulles)



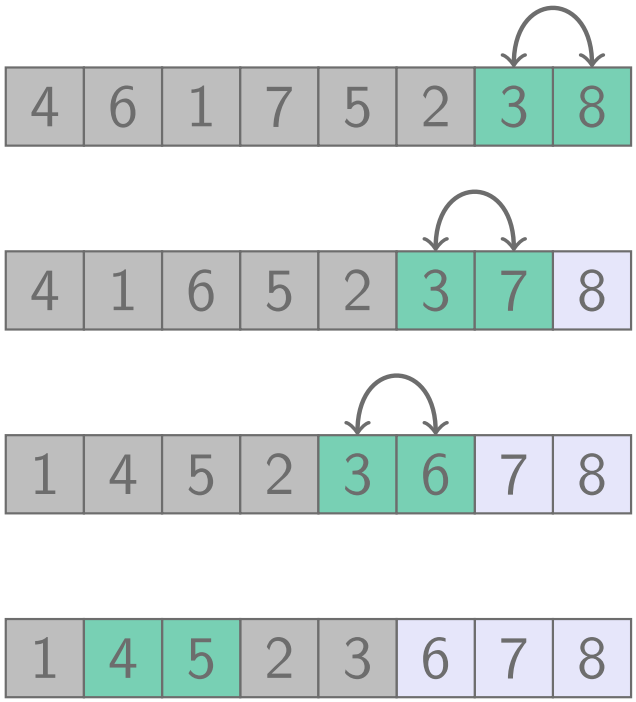
Tri par propagation (ou tri à bulles)



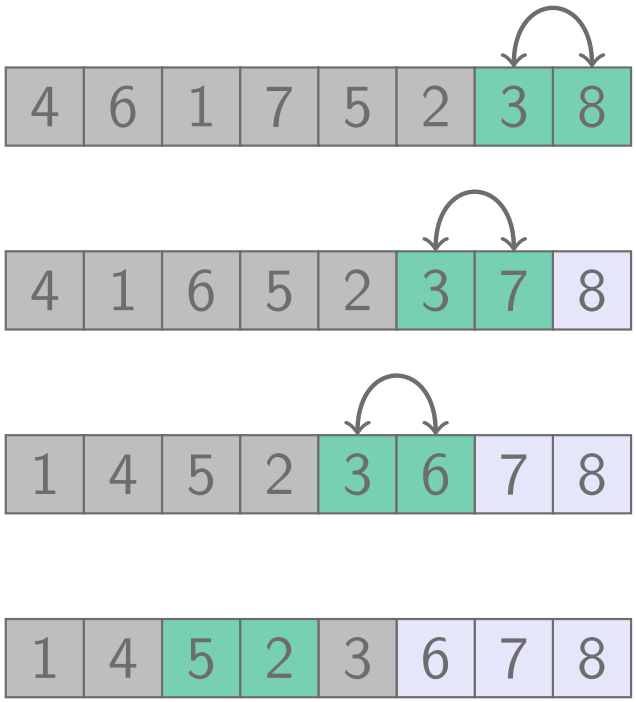
Tri par propagation (ou tri à bulles)



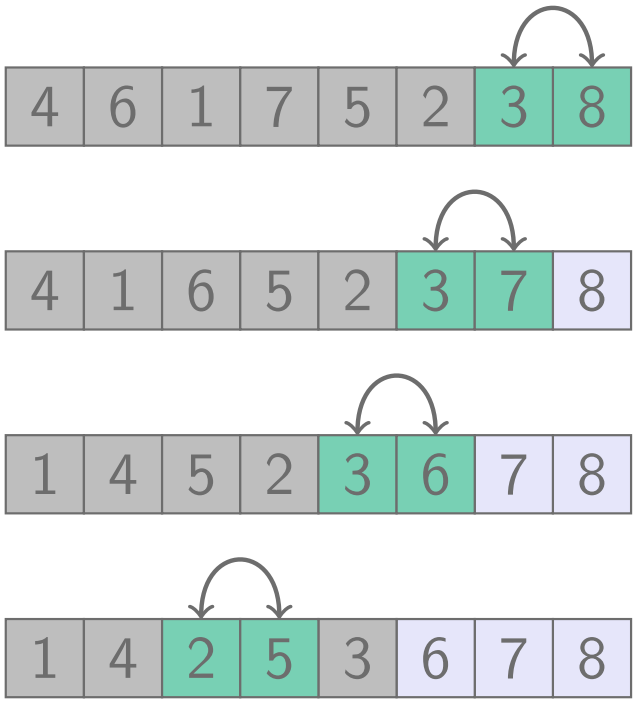
Tri par propagation (ou tri à bulles)



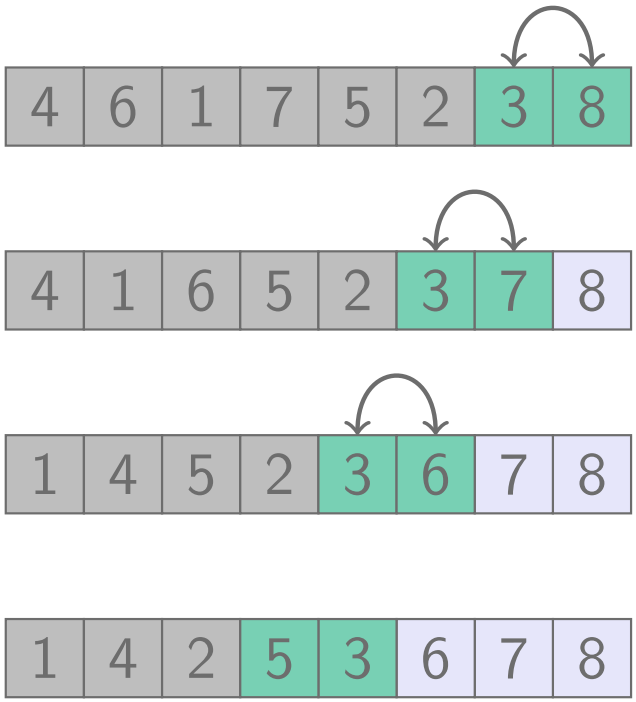
Tri par propagation (ou tri à bulles)



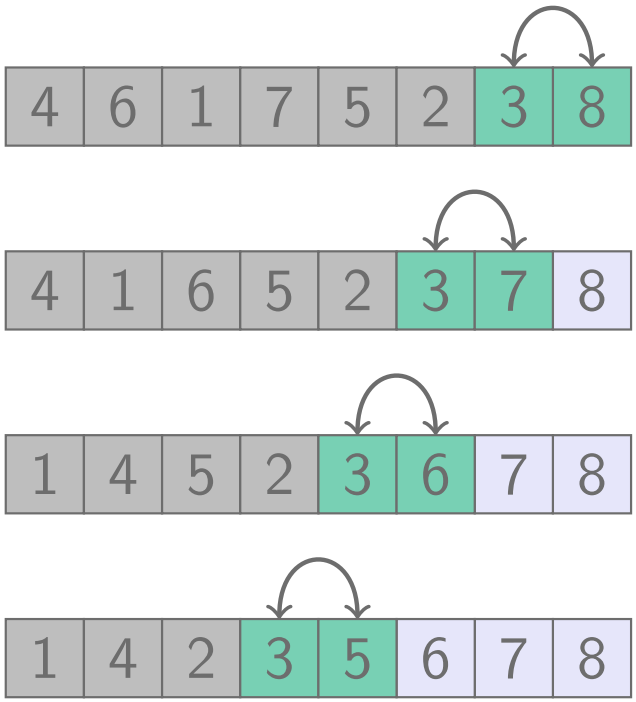
Tri par propagation (ou tri à bulles)



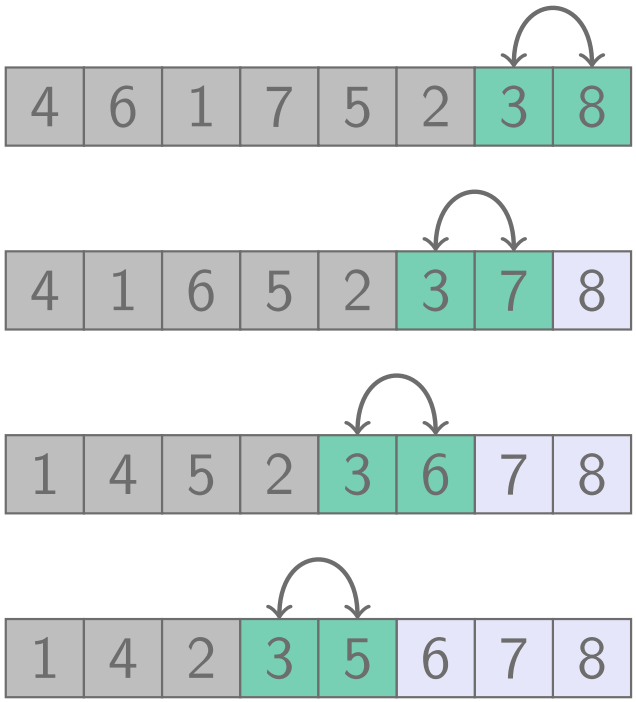
Tri par propagation (ou tri à bulles)



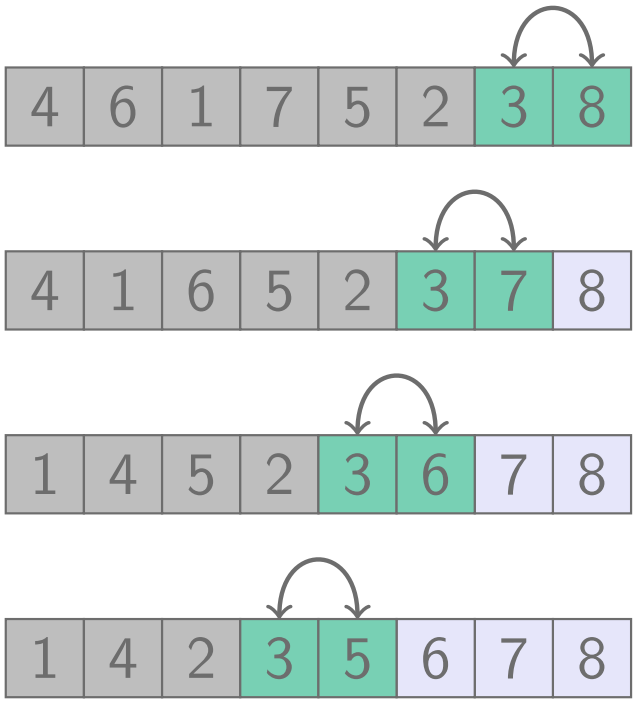
Tri par propagation (ou tri à bulles)



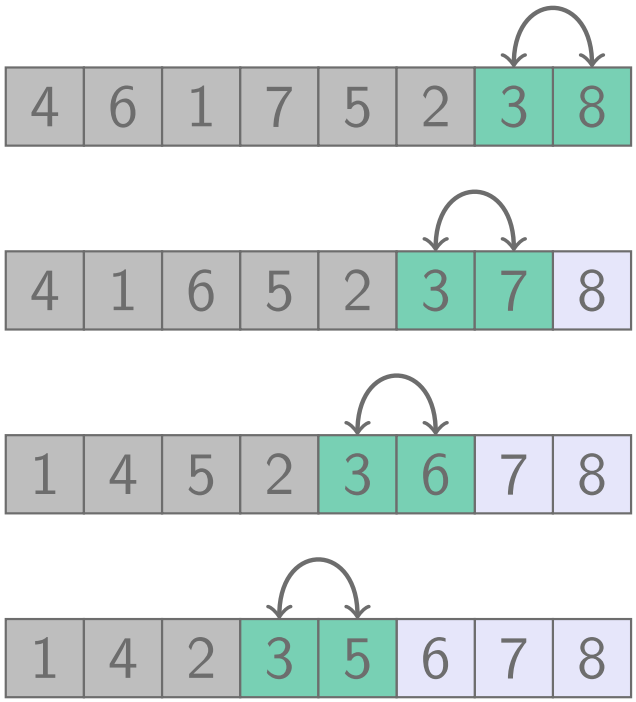
Tri par propagation (ou tri à bulles)

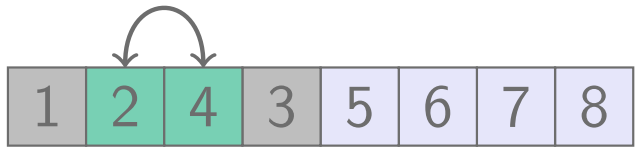


Tri par propagation (ou tri à bulles)

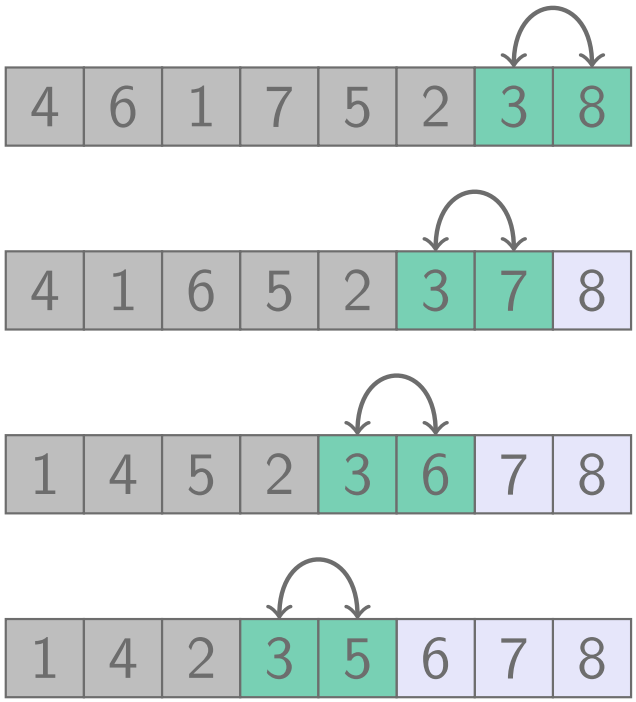


Tri par propagation (ou tri à bulles)

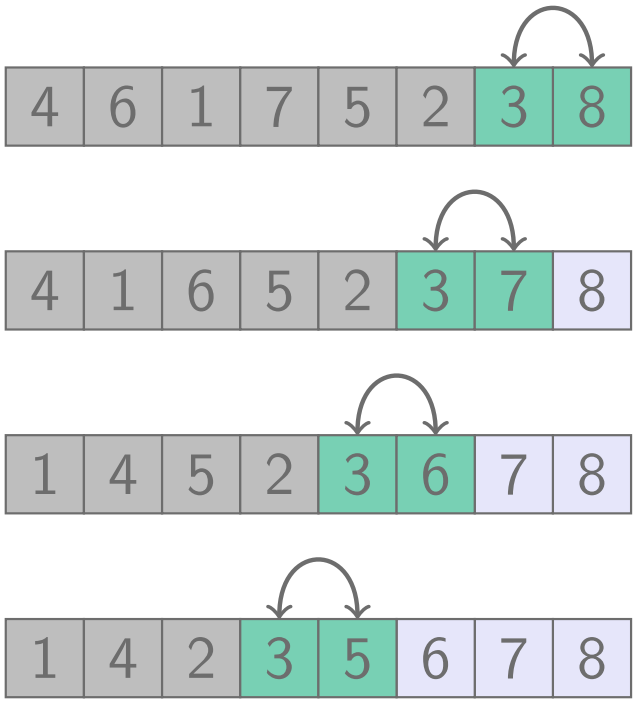


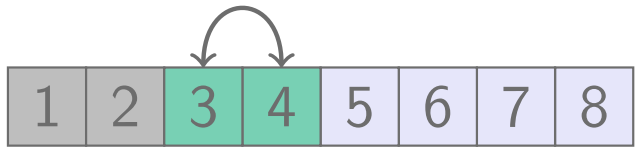


Tri par propagation (ou tri à bulles)

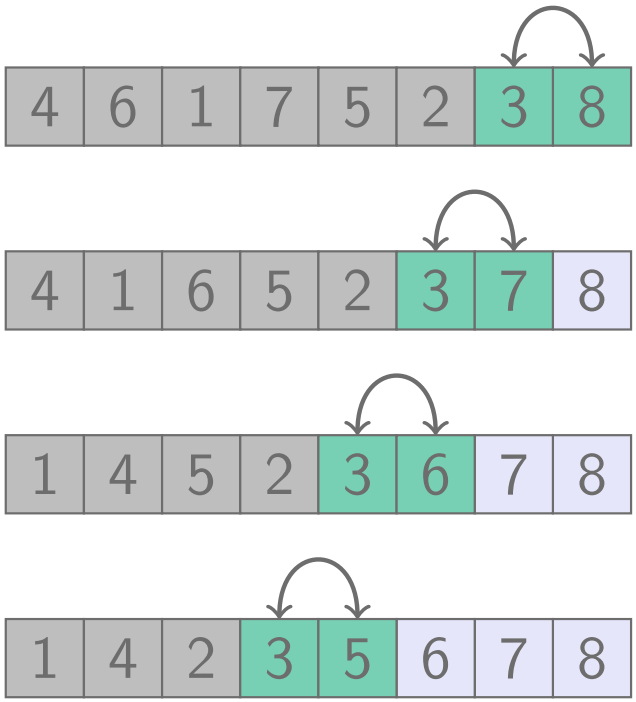


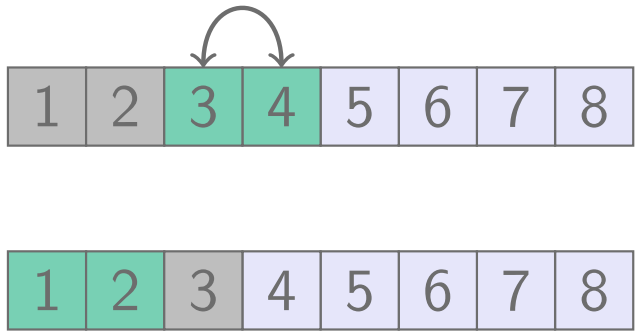
Tri par propagation (ou tri à bulles)



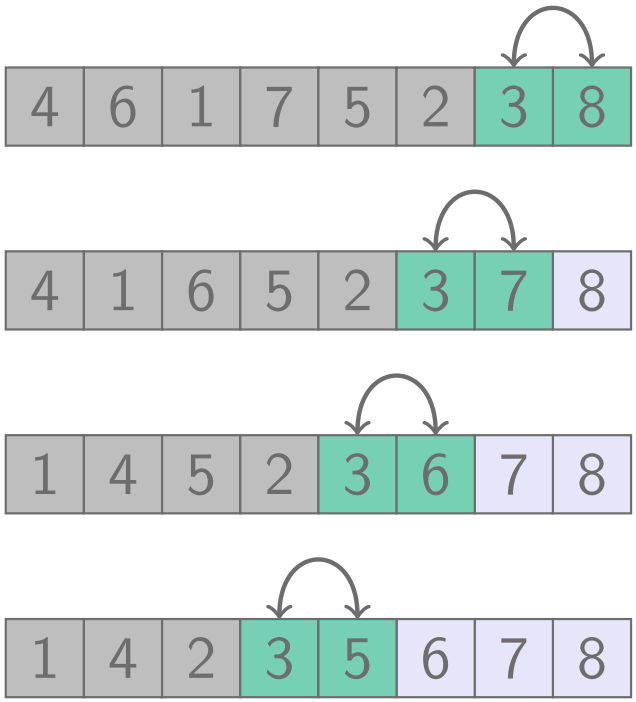


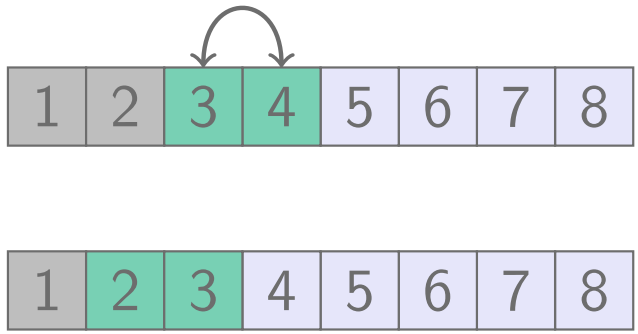
Tri par propagation (ou tri à bulles)



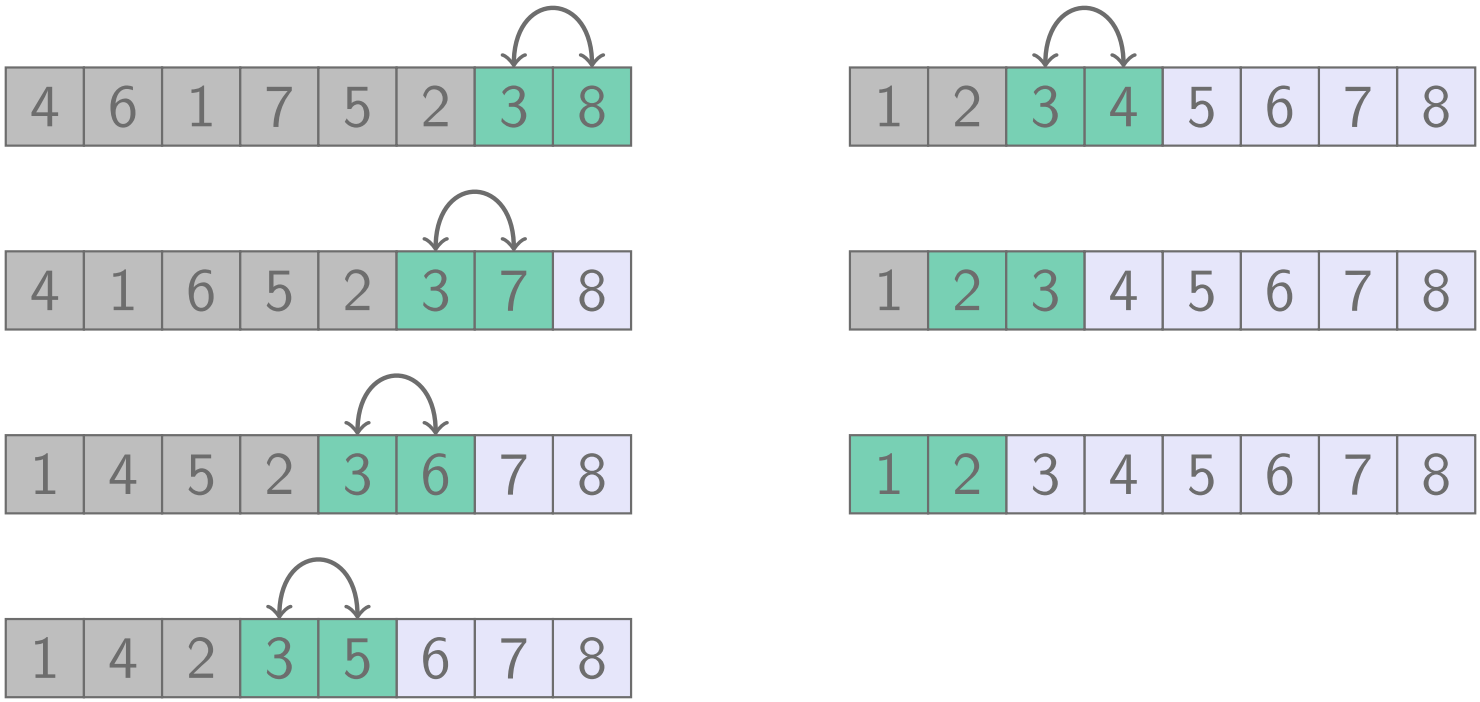


Tri par propagation (ou tri à bulles)

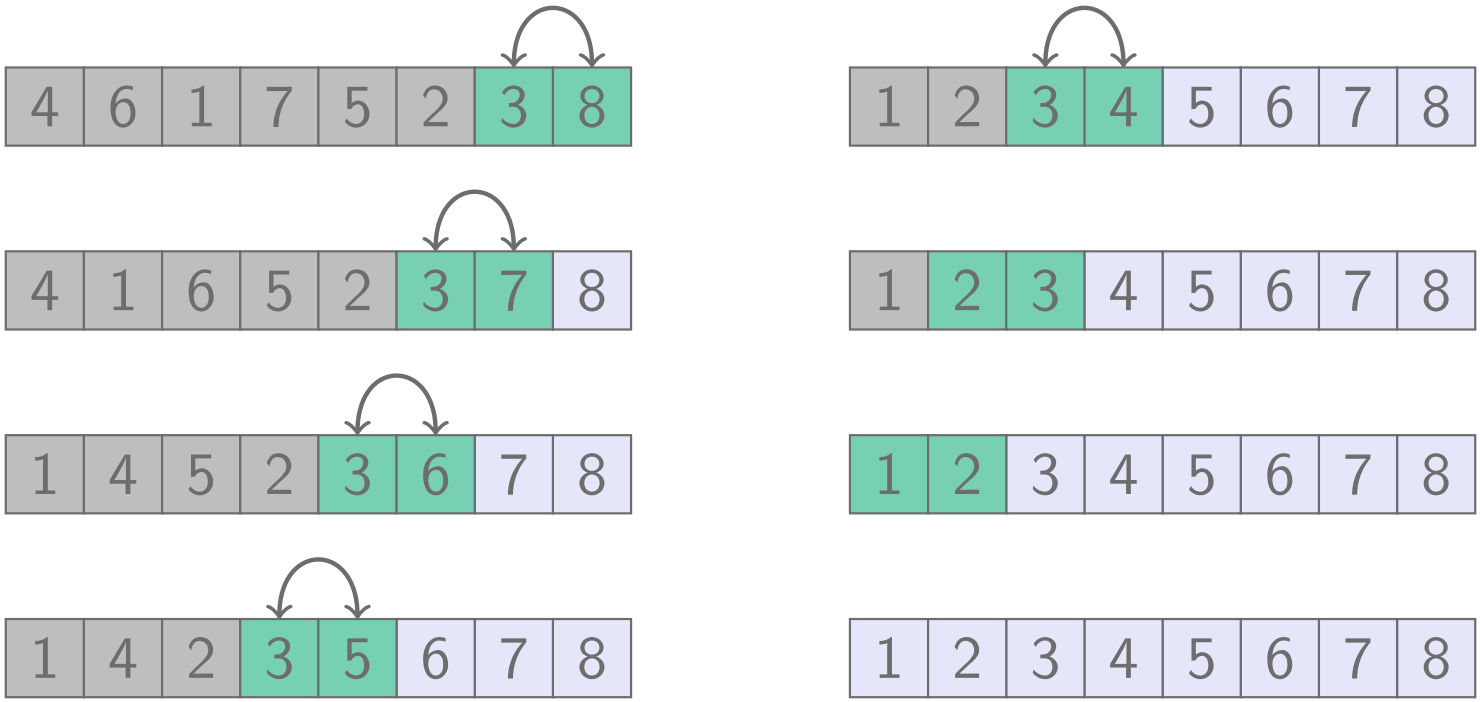




Tri par propagation (ou tri à bulles)



Tri par propagation (ou tri à bulles)



Tri par propagation (ou tri à bulles)

```
1  ALGORITHME TriBulles(T):
2  DONNEES
3    T: tableau d'entier de taille n
4  VARIABLES:
5    d,i: entiers
6  DEBUT
7    d←n
8    TQ d > 1 FAIRE
9      i←1
10     TQ i < d FAIRE
11       SI T[i] > T[i+1] ALORS
12         Swap(T,i,i+1)
13       FSI
14       i←i+1
15     FTQ
16   d←d−1
17   FTQ
18 FIN
```

- Arrêt : deux boucles imbriquées à incrément constant
- Validité : $(T[d+1:n]$ est trié et $t[d+1] \geq max(T[1:d]))$ est un invariant
- Complexité : $\Theta(n^2)$

Tri par propagation (ou tri à bulles)

- Si aucun échange n'est effectué alors le tableau est trié.
- C'est un moyen simple pour savoir quand arrêter le travail de l'algorithme.
- On améliore ainsi la complexité dans le meilleur cas.

Tri par propagation (ou tri à bulles)

```
1  ALGORITHME TriBulles(T):
2  DEBUT
3     d←n
4     echange←VRAI
5     TQ echnage = VRAI FAIRE
6        i←1
7        echange←FAUX
8        TQ i < d FAIRE
9           SI T[i] > T[i+1] ALORS
10             Swap(T,i,i+1)
11             echange←VRAI
12          FSI
13          i←i+1
14       FTQ
15    d←d−1
16    FTQ
17 FIN
```

Tri par propagation (ou tri à bulles)

```
1  ALGORITHME TriBulles(T):
2  DEBUT
3     d←n
4     echange←VRAI
5     TQ echnage = VRAI FAIRE
6        i←1
7        echange←FAUX
8        TQ i < d FAIRE
9           SI T[i] > T[i+1] ALORS
10             Swap(T,i,i+1)
11             echange←VRAI
12          FSI
13          i←i+1
14       FTQ
15    d←d−1
16    FTQ
17 FIN
```

- Meilleur cas : $\Theta(n)$
- Pire cas : $\Theta(n^2)$
- Complexité : $O(n^2)$

Tri par insertion

Idée générale :

- Tri utiliser naturellement pour ranger des cartes ;
- on parcours le tableau de droite à gauche et on range chaque élément à sa place parmis les éléments précédents.

Tri par insertion

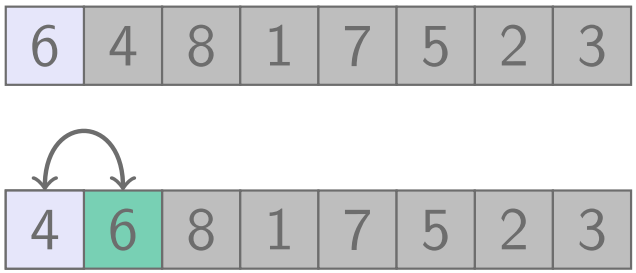
6	4	8	1	7	5	2	3

Tri par insertion

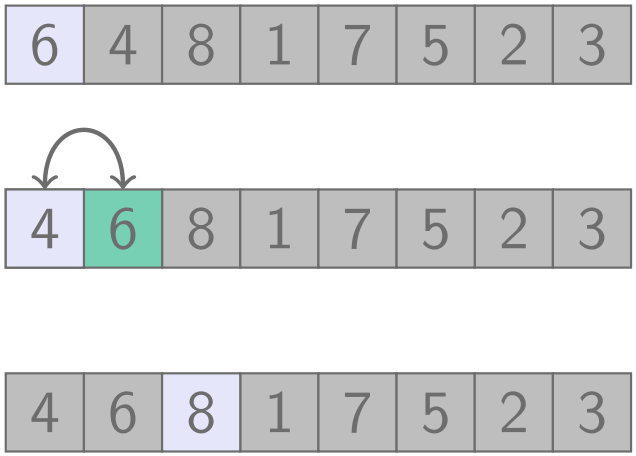
Tri par insertion

Tri par insertion

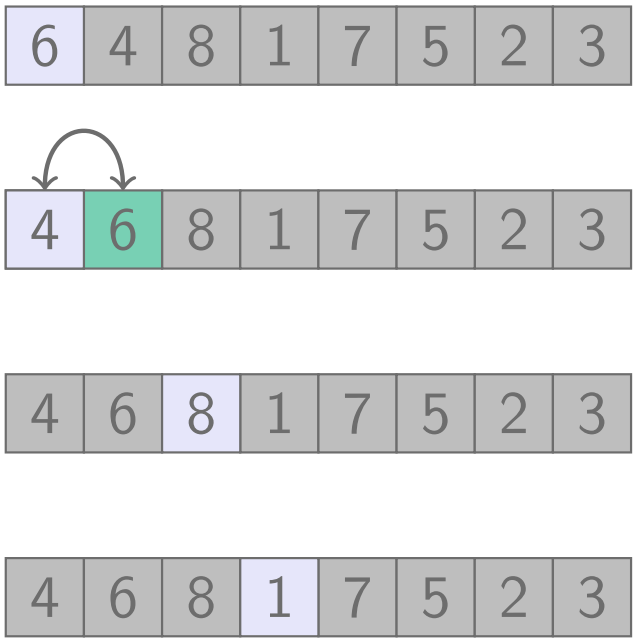
Tri par insertion



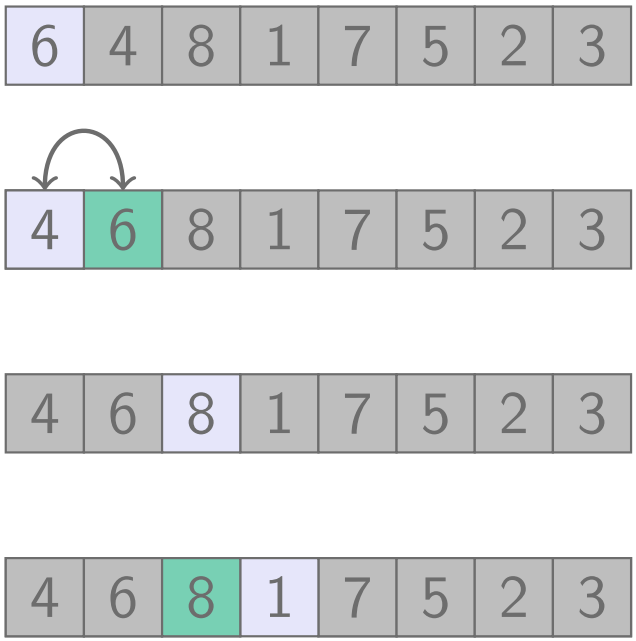
Tri par insertion



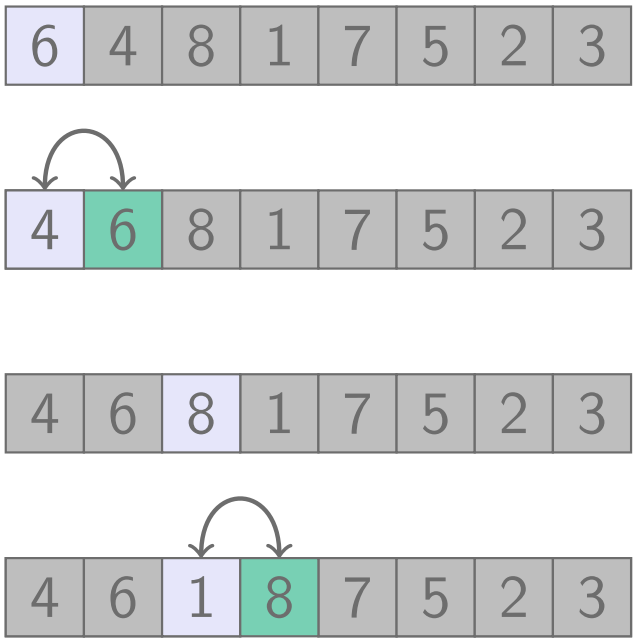
Tri par insertion



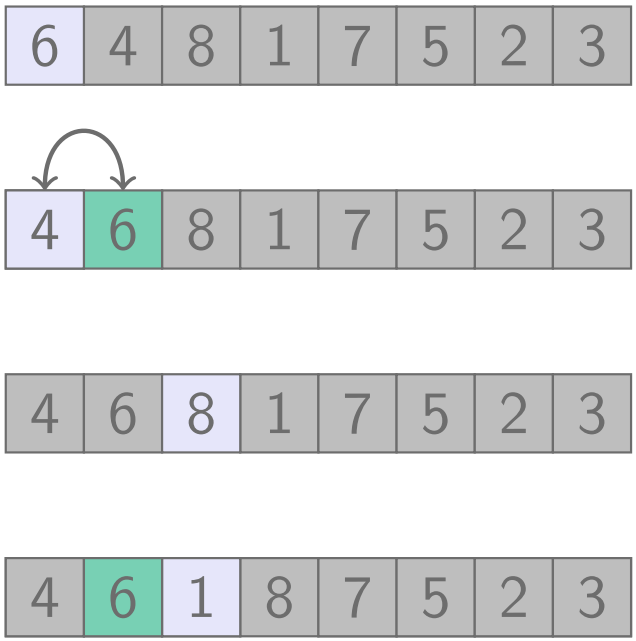
Tri par insertion



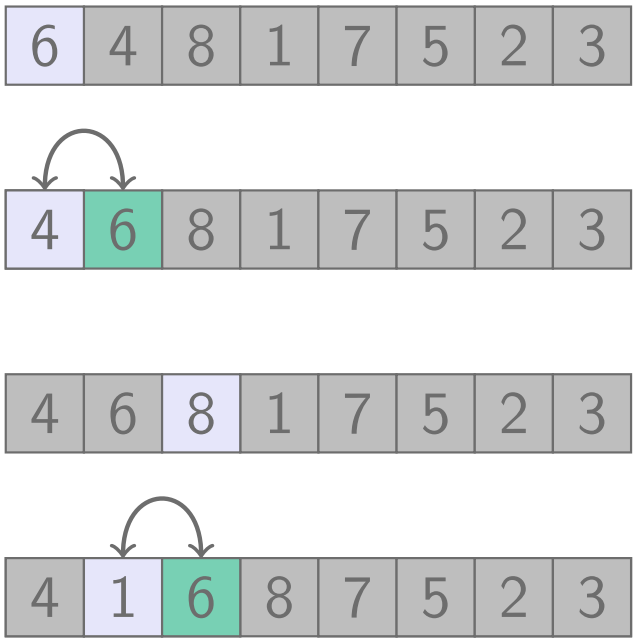
Tri par insertion



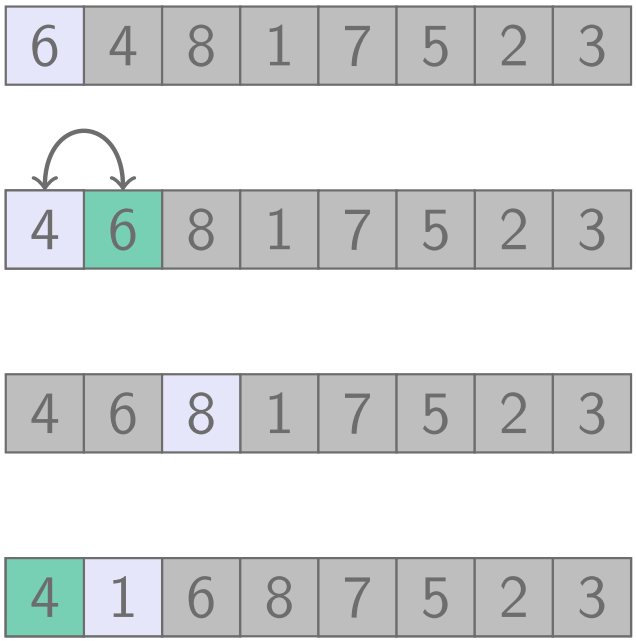
Tri par insertion



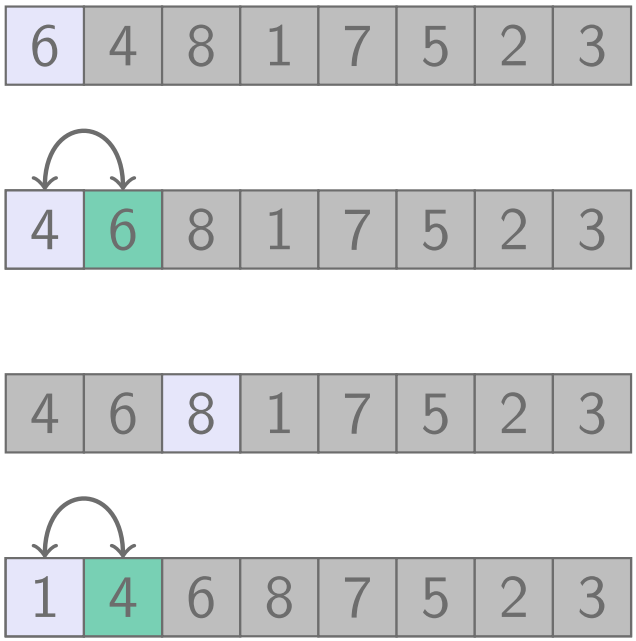
Tri par insertion



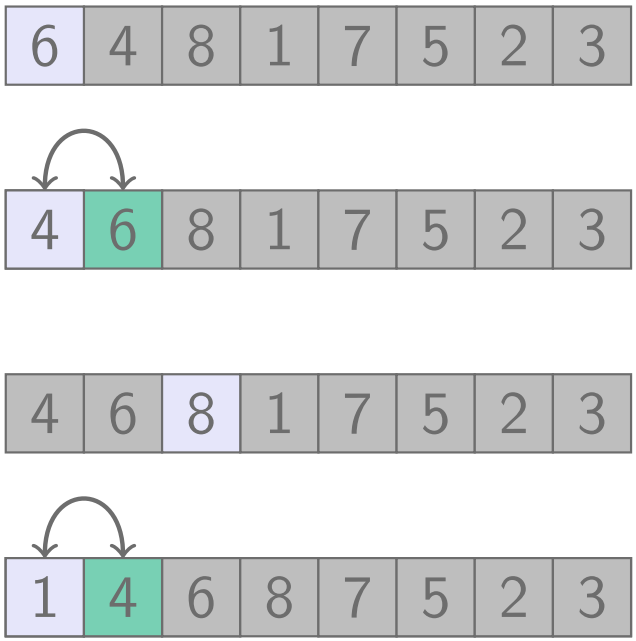
Tri par insertion

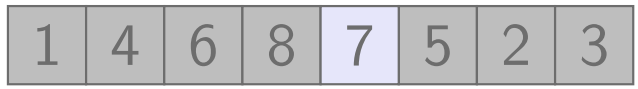


Tri par insertion

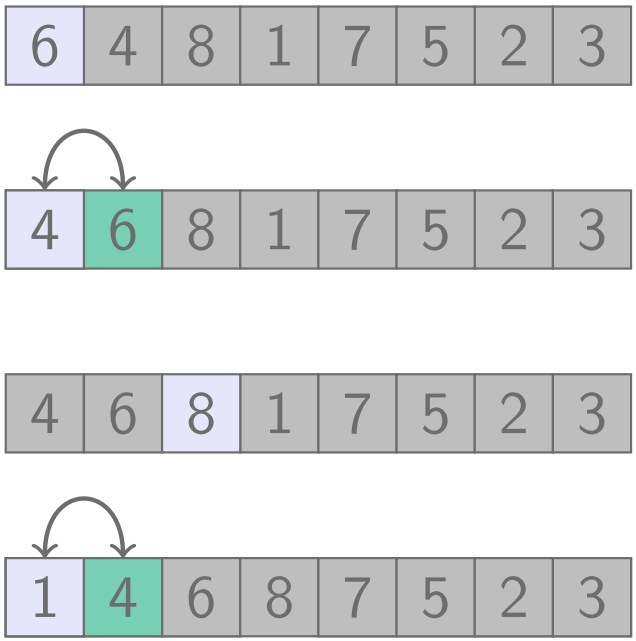


Tri par insertion

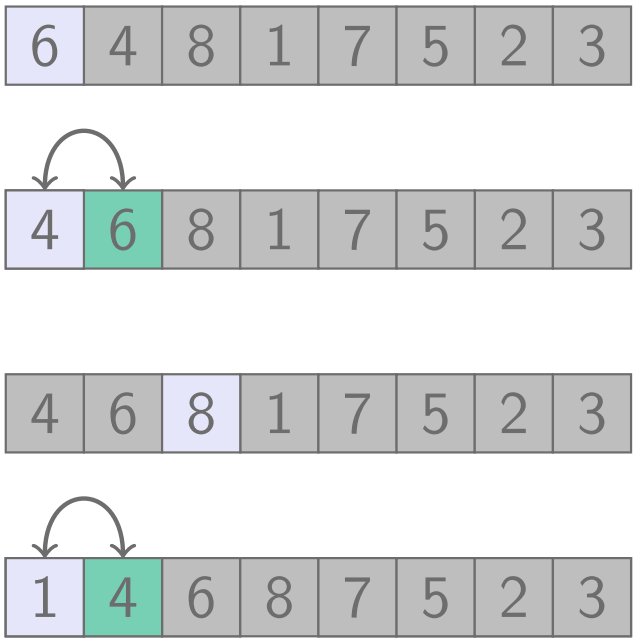


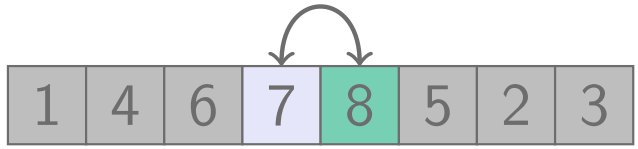


Tri par insertion

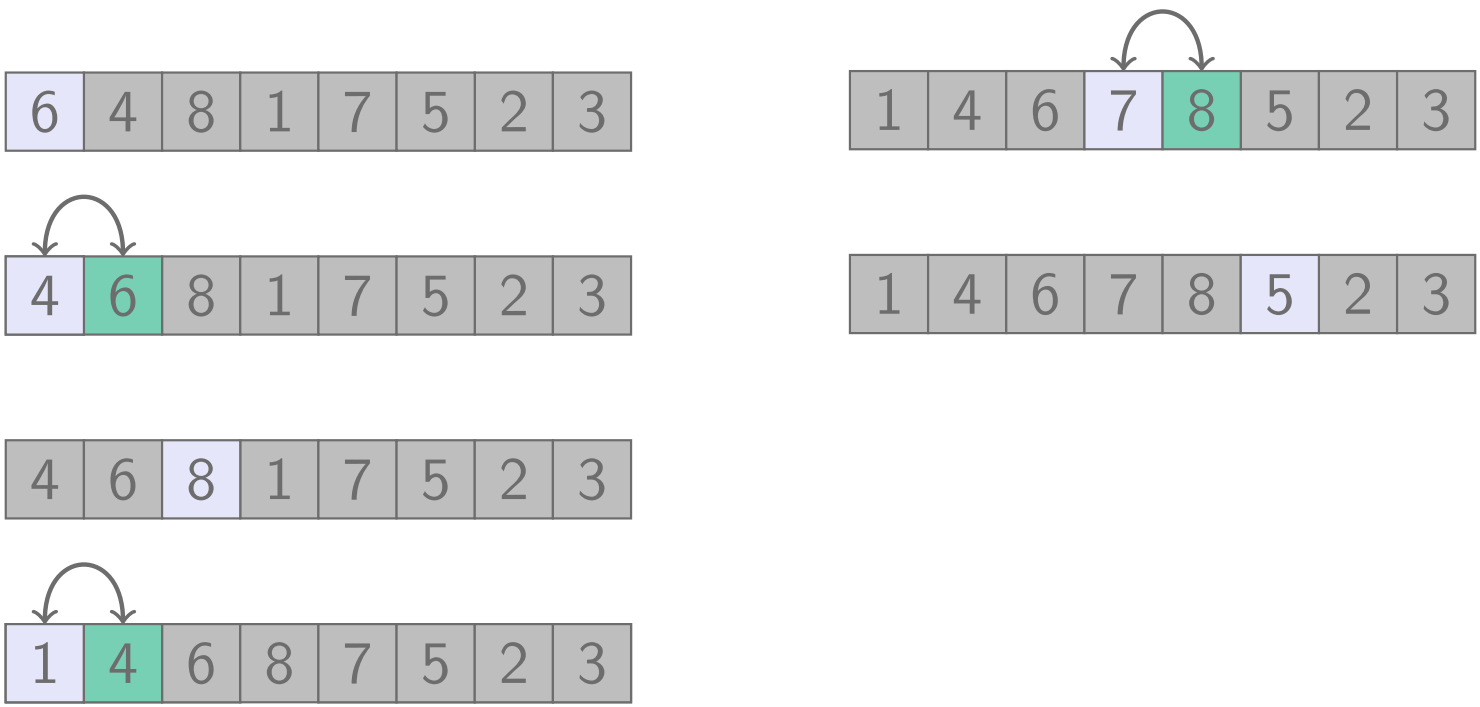


Tri par insertion

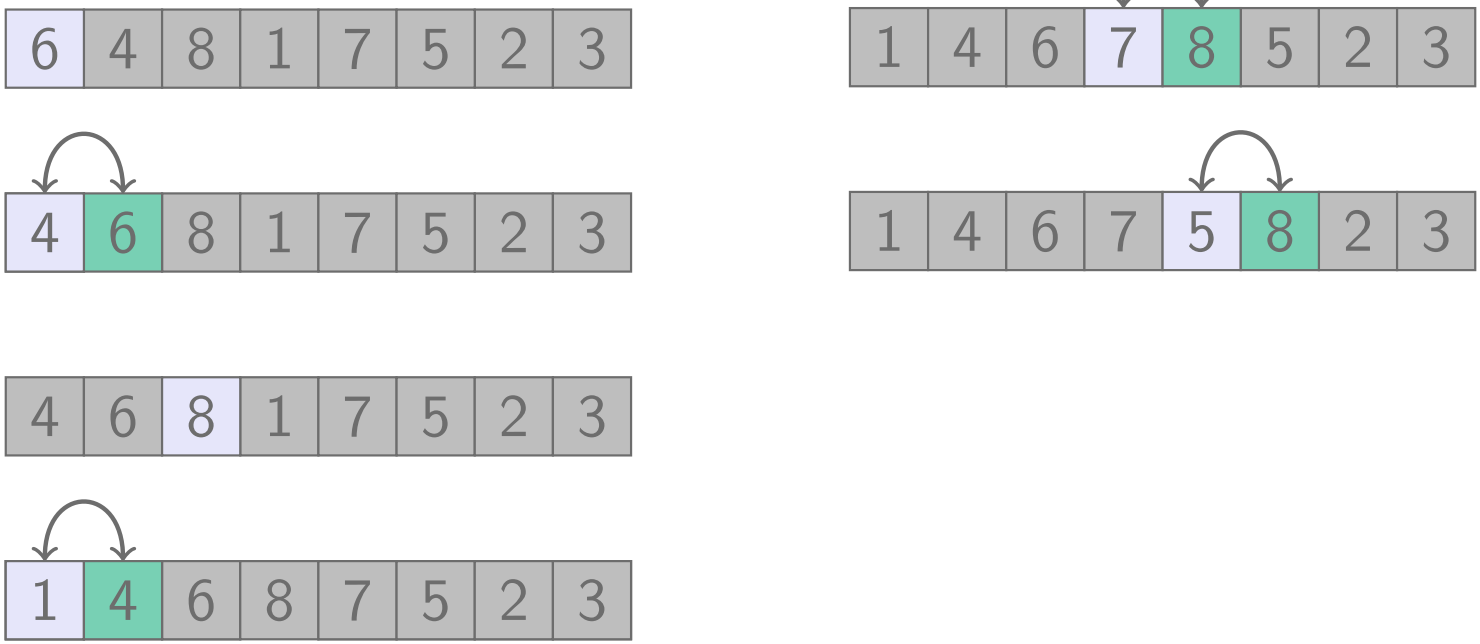




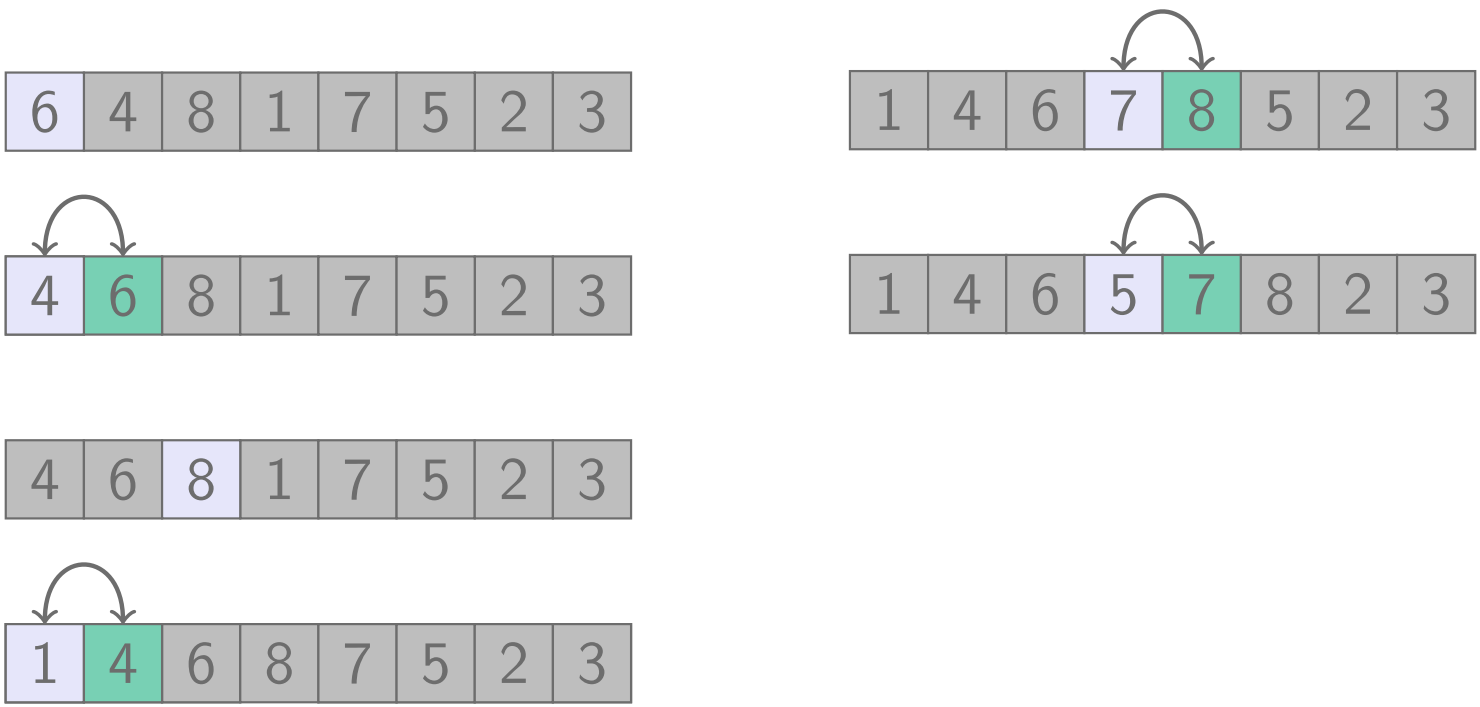
Tri par insertion



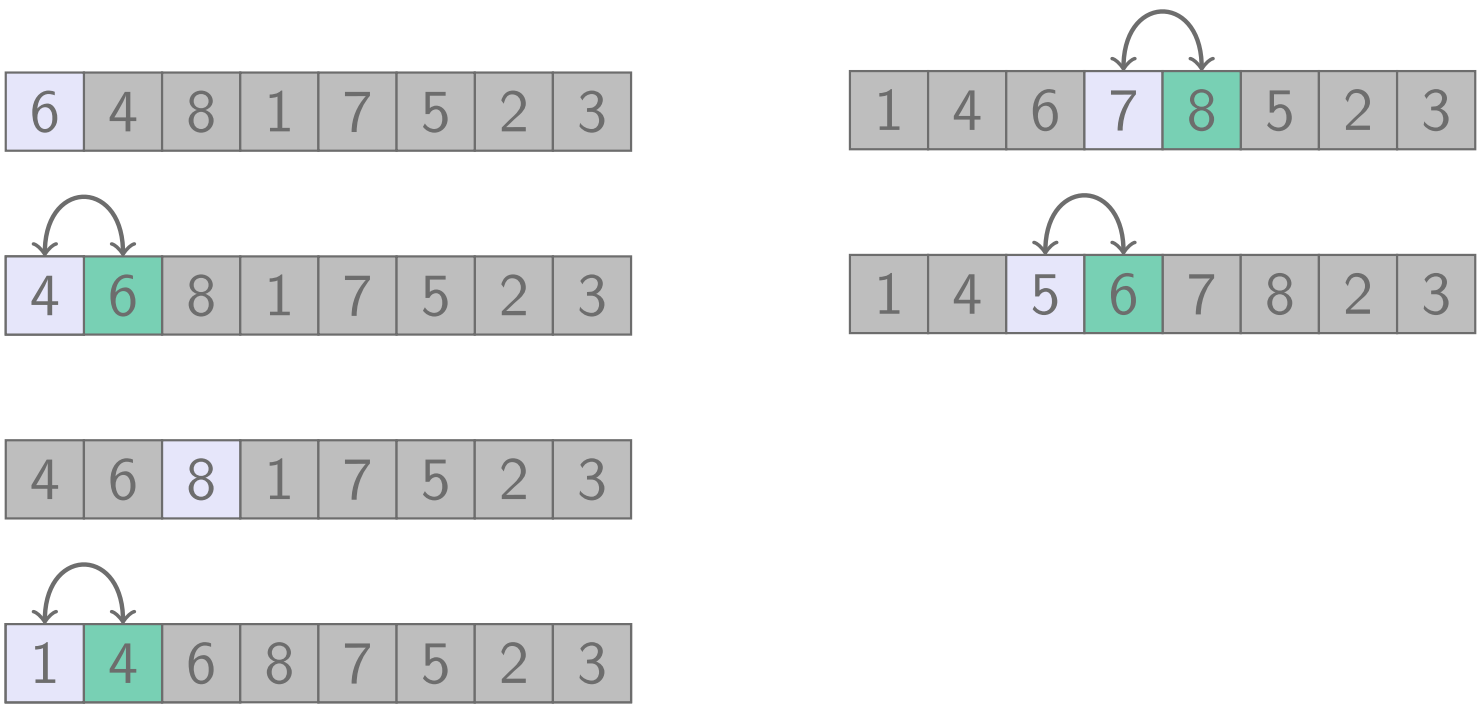
Tri par insertion



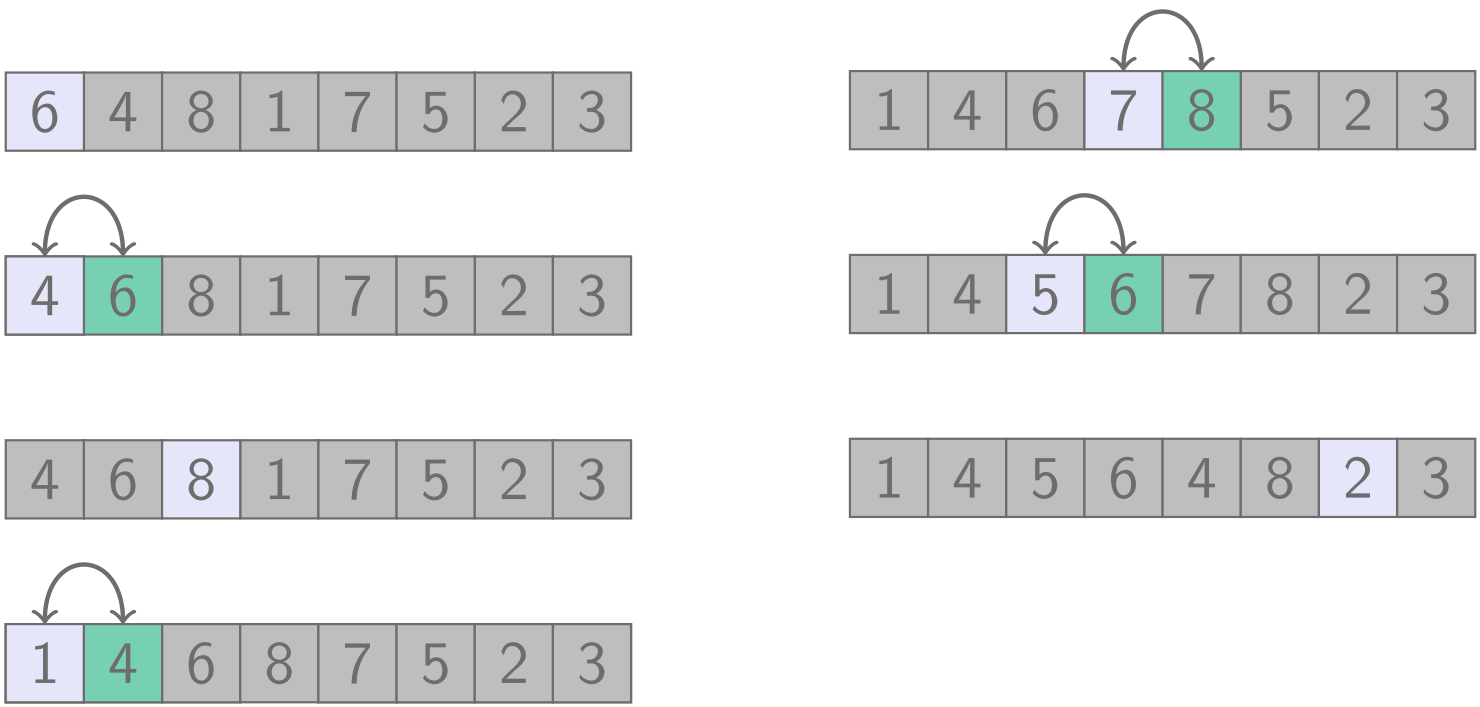
Tri par insertion



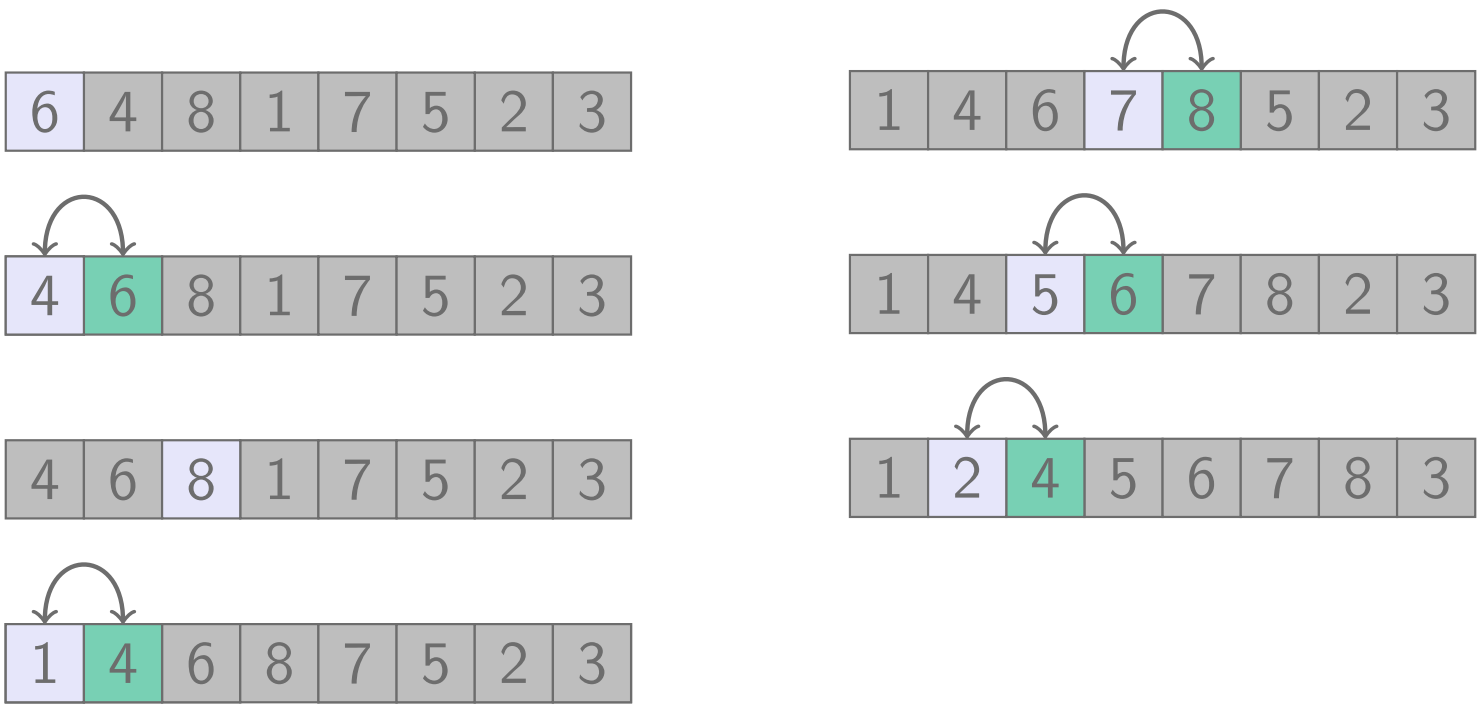
Tri par insertion



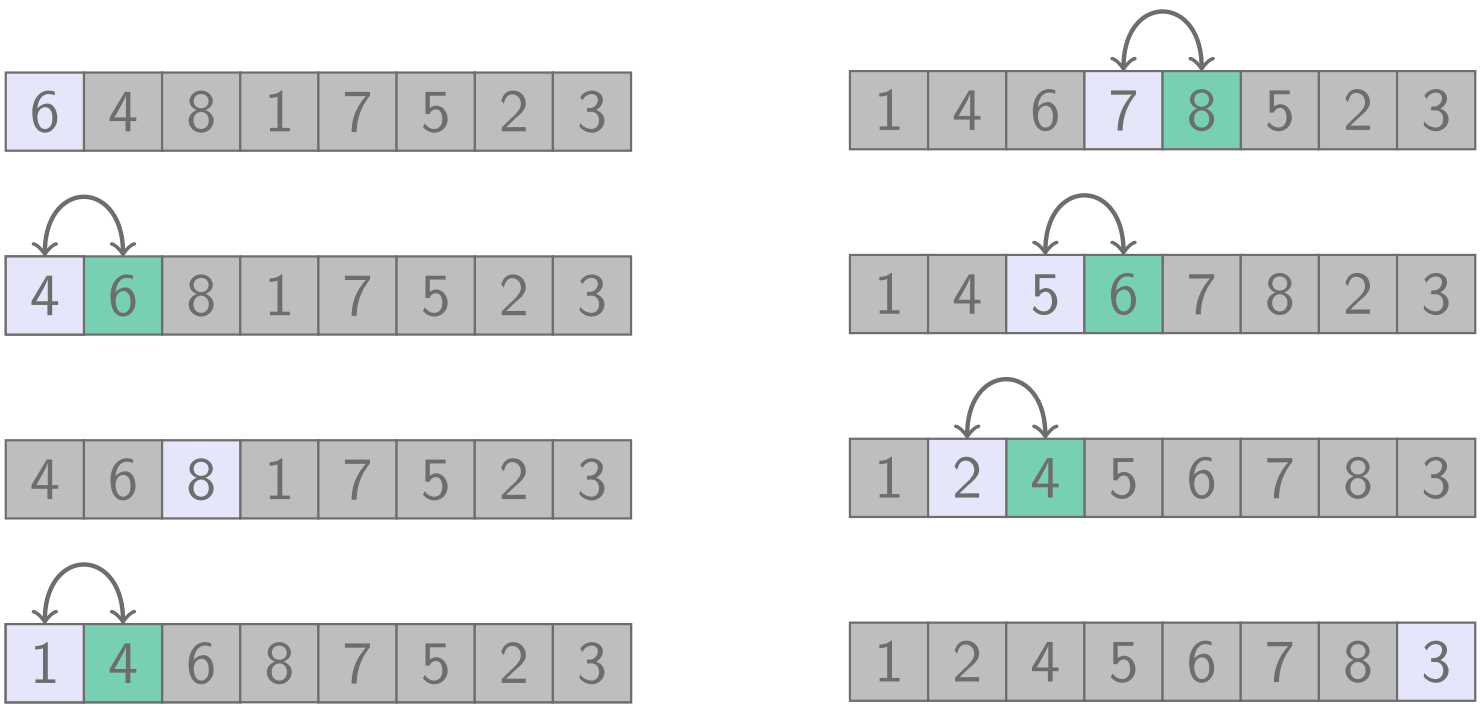
Tri par insertion



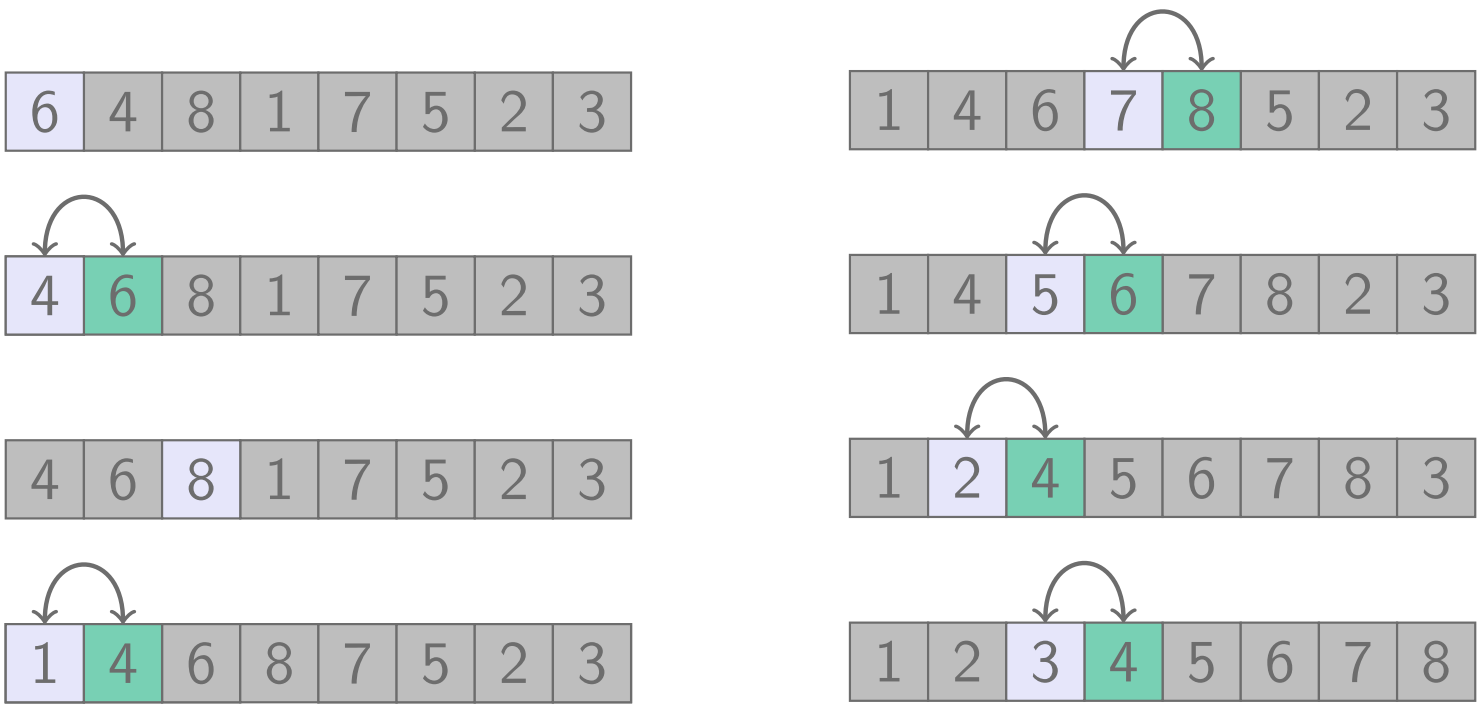
Tri par insertion



Tri par insertion



Tri par insertion



Tri par insertion

```
1  ALGORITHME TriInsertion(T):
2  DONNEES
3    T: tableau d'entier de taille n
4  VARIABLES:
5    i,j: entiers
6  DEBUT
7    i←2
8    TQ i ≤ n FAIRE
9      j←i
10     TQ j > 1 ET T[j−1]>T[j] FAIRE
11       Swap(T,j,j−1)
12       j←j−1
13     FTQ
14     i←i+1
15   FTQ
16 FIN
```

Tri par insertion

```
1  ALGORITHME TriInsertion(T):
2  DONNEES
3    T: tableau d'entier de taille n
4  VARIABLES:
5    i,j: entiers
6  DEBUT
7    i←2
8    TQ i ≤ n FAIRE
9      j←i
10     TQ j > 1 ET T[j−1]>T[j] FAIRE
11       Swap(T,j,j−1)
12       j←j−1
13     FTQ
14     i←i+1
15   FTQ
16 FIN
```

- Arrêt : deux boucles imbriquées à incrément constant
- Validité : ($T[1:i-1]$ est trié) est un invariant
- Complexité : $O(n^2)$

Tri comparatif optimal

- Les algorithmes de tri en $\Theta(n^2)$ sont considérés comme lents
- Il est heureusement possible d'obtenir des algorithmes de tris de complextié

$$\Theta(n \log(n)).$$

- Il est démontrable qu'il n'exite aucun algorithme comparatif de complexité strictement inférieure résolvant le problème général du tri

Application du Tri

Problème : élément d'occurence maximale

Entrée : tableau d'entiers T de taille n
Sortie : l'élément ayant le plus grand nombre d'occurence dans le tableau

- Approche naïve : compter le nombre d'occurences de chaque élément du tableau
- complexité : $\Theta(n^2)$

Application du Tri

6	4	1	1	6	1	2	3	2	1	6

Application du Tri

Application du Tri

Application du Tri

Application du Tri

Application du Tri

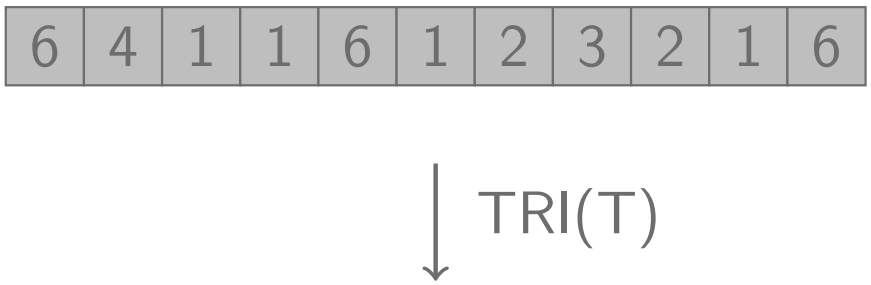
Application du Tri

Si le tableau est trié, le problème devient plus simple

6	4	1	1	6	1	2	3	2	1	6

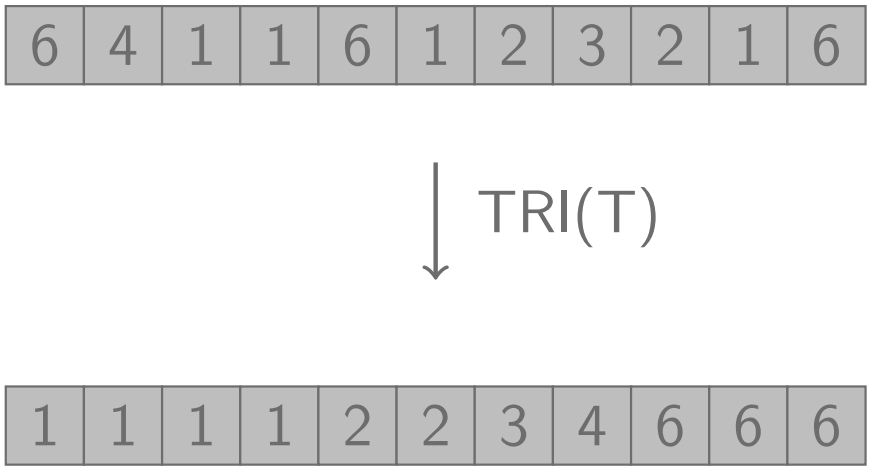
Application du Tri

Si le tableau est trié, le problème devient plus simple



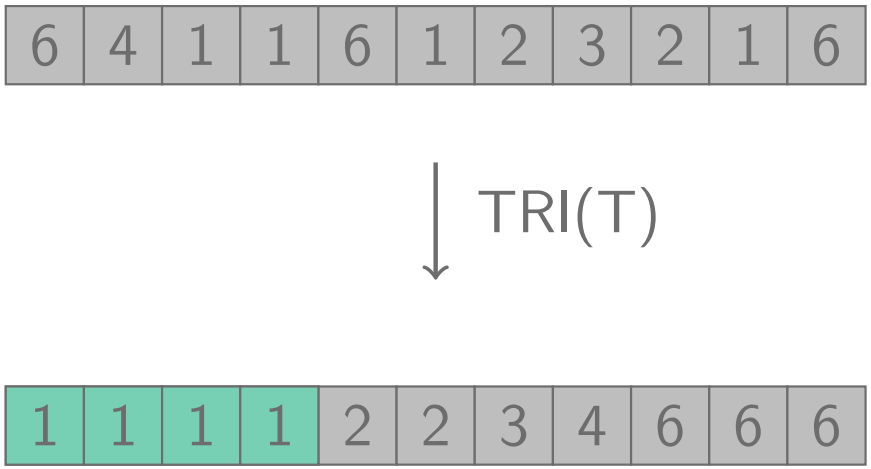
Application du Tri

Si le tableau est trié, le problème devient plus simple



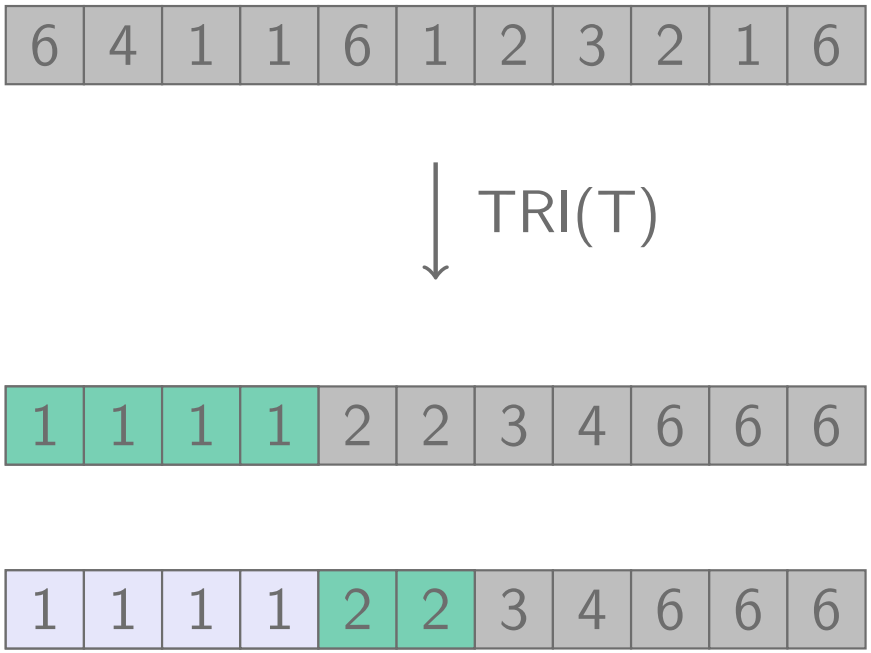
Application du Tri

Si le tableau est trié, le problème devient plus simple



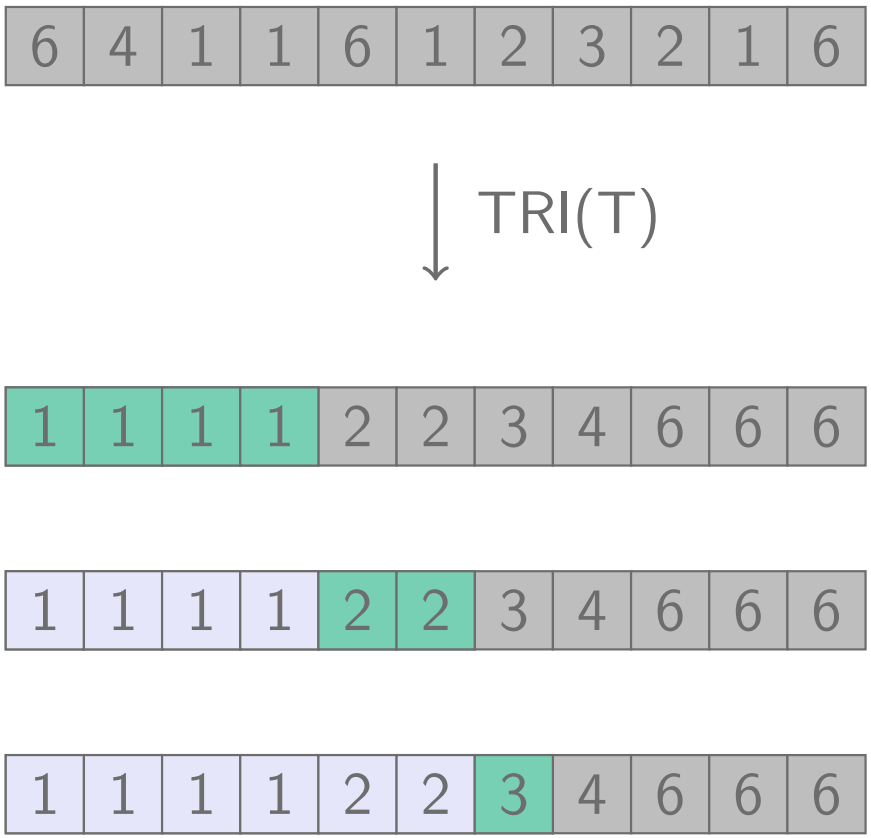
Application du Tri

Si le tableau est trié, le problème devient plus simple



Application du Tri

Si le tableau est trié, le problème devient plus simple



Application du Tri

```
1  ALGORITHME OCCMAX(T):
2  DEBUT
3     TRI(T)
4     imax,occmax ← 0,0
5     i,occ ← 2,1
6     TQ i ≤ n FAIRE
7        SI T[i-1] = T[i] ALORS
8           occ ← occ+1
9        SINON
10          SI occ > occmax ALORS
11             occmax ← occ
12             imax ← i-1
13          FSI
14          occ ← 1
15       FSI
16       i ← i+1
17    FTQ
18    RENVOYER imax
19 FIN
```

- Complexité :

$$C_{\text{OMAX}}(n) = \Theta(n) + C_{TRI}(n)$$

- Avec un tri optimal

$$C_{\text{OMAX}}(n) = \Theta(n \log(n))$$

Tri comptage

- Avec des hypothèses supplémentaires il est possible d'obtenir des algorithmes de tri de complexité inférieure à $\Theta(n \log(n))$
- En considérant par exemple que
  1. les éléments du tableaux T sont bornés
  2. on peut écrire directement dans les cases du tableau (et non pas seulement permuter des éléments)

  on peut définir un problème de tri plus simple que le problème général :

Tri comptage

Problème : Tri borné

Entrée : tableau d'entiers T de taille n dont les éléments sont bornés entre 1 et k
Sortie : une réécriture des éléments de T telle que

$$T[1] \leq T[2] \leq \cdots \leq T[n].$$

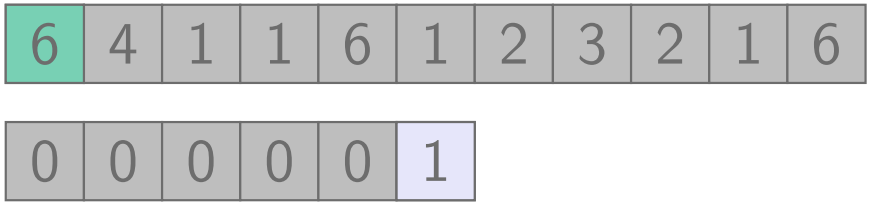
- L'idée principale pour résoudre ce problème est de compter le nombre d'occurences de chaque nombre entre 1 et k dans un tableau auxiliaire.

Tri comptage

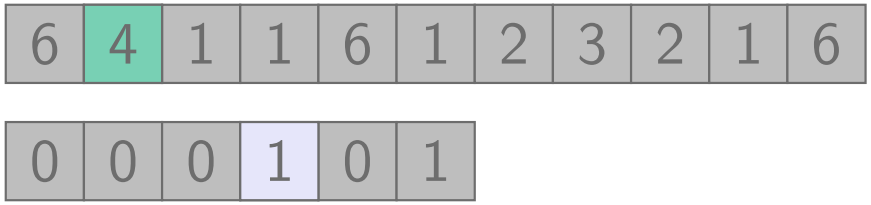
6	4	1	1	6	1	2	3	2	1	6

0	0	0	0	0	0

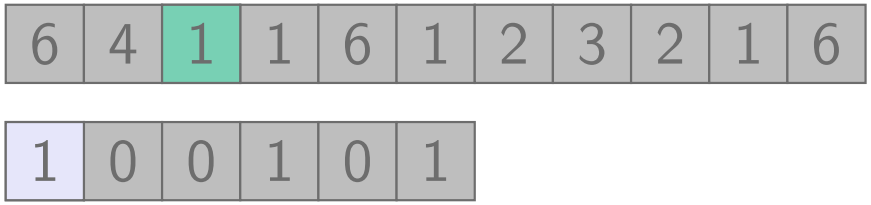
Tri comptage



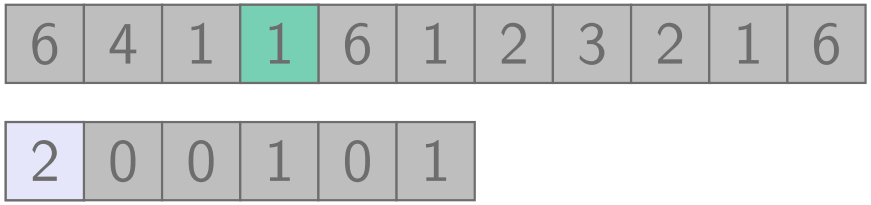
Tri comptage



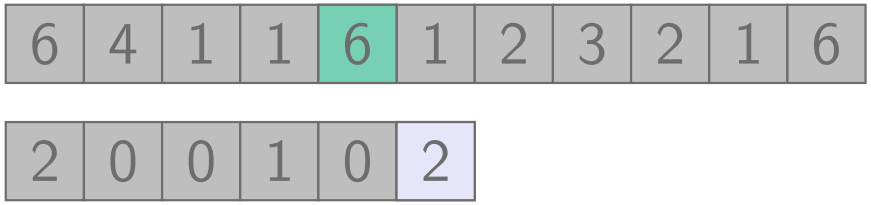
Tri comptage



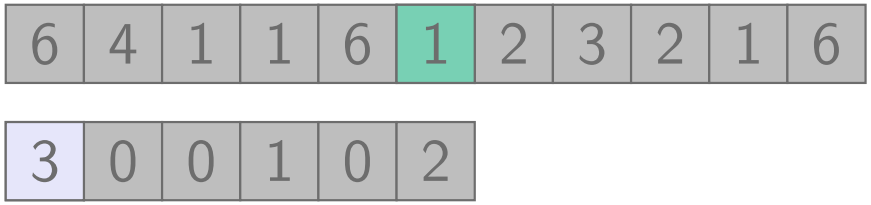
Tri comptage



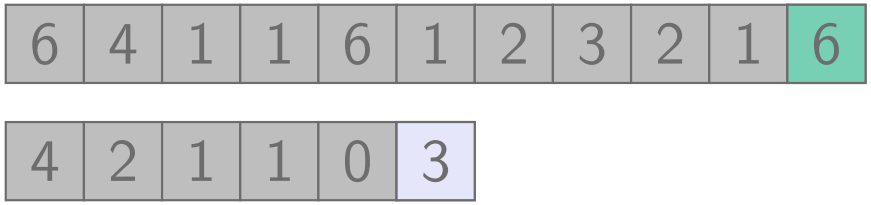
Tri comptage



Tri comptage



Tri comptage



Tri comptage

6	4	1	1	6	1	2	3	2	1	6

4	2	1	1	0	3

↓ Réécriture

Tri comptage

6	4	1	1	6	1	2	3	2	1	6

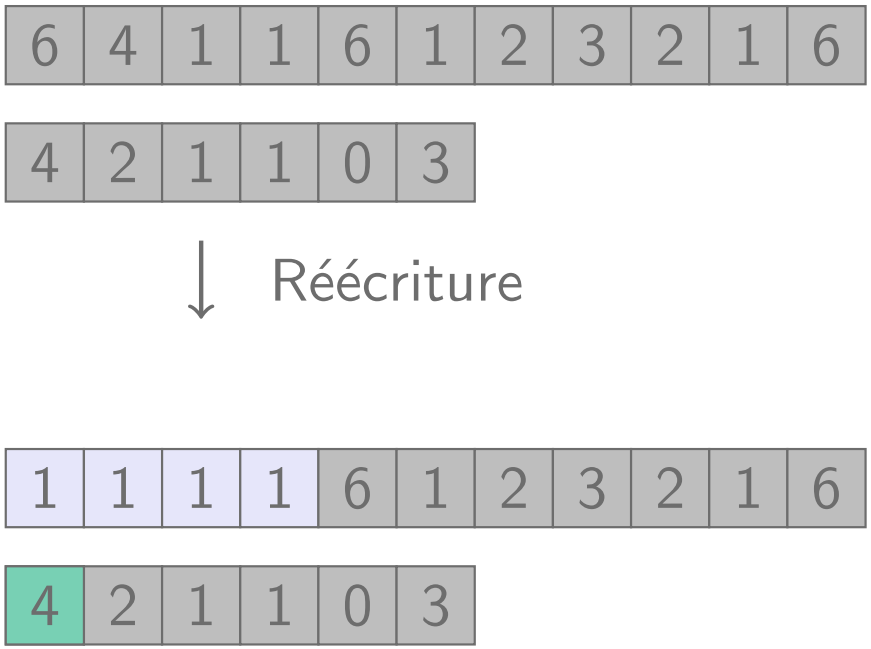
4	2	1	1	0	3

↓ Réécriture

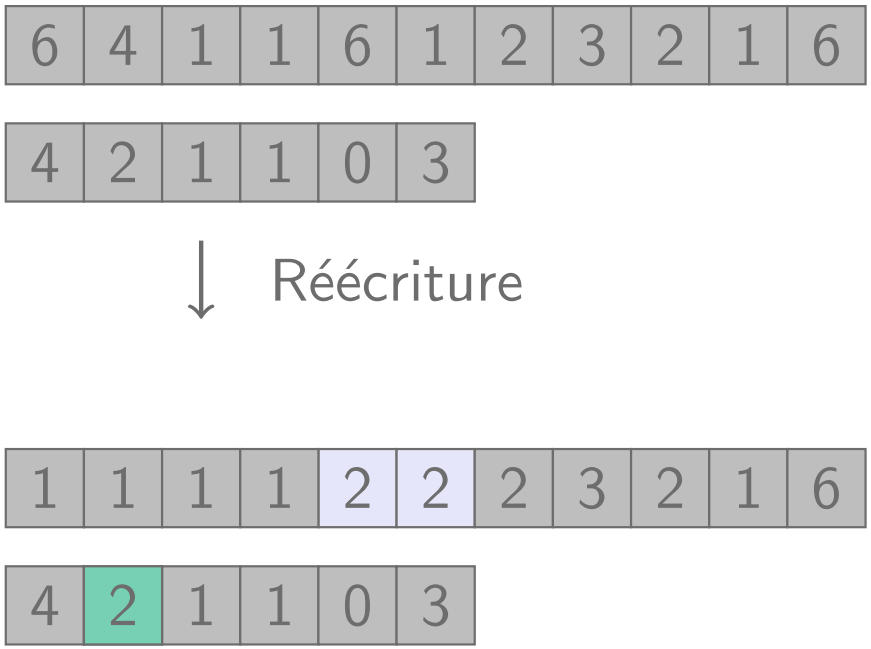
6	4	1	1	6	1	2	3	2	1	6

4	2	1	1	0	3

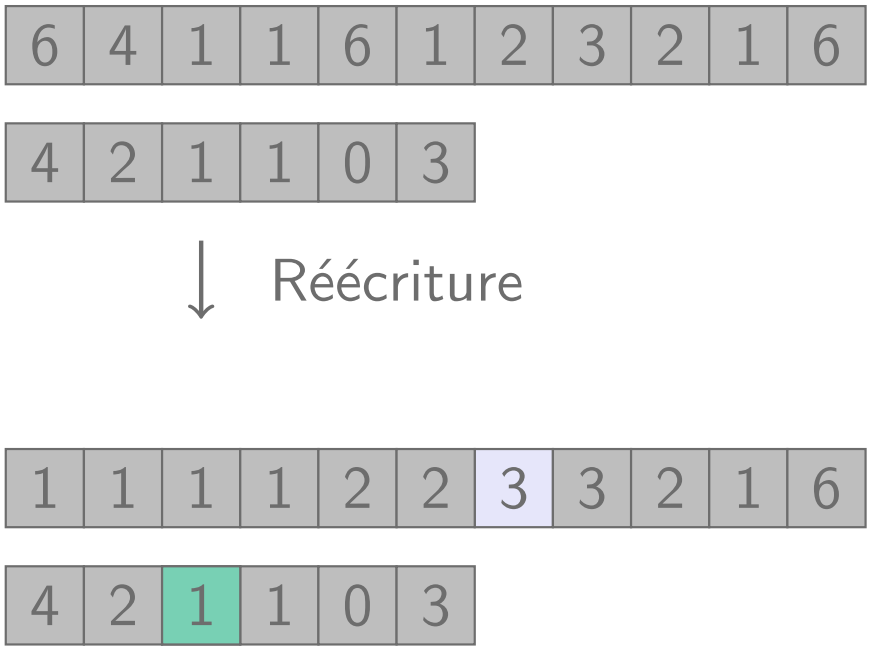
Tri comptage



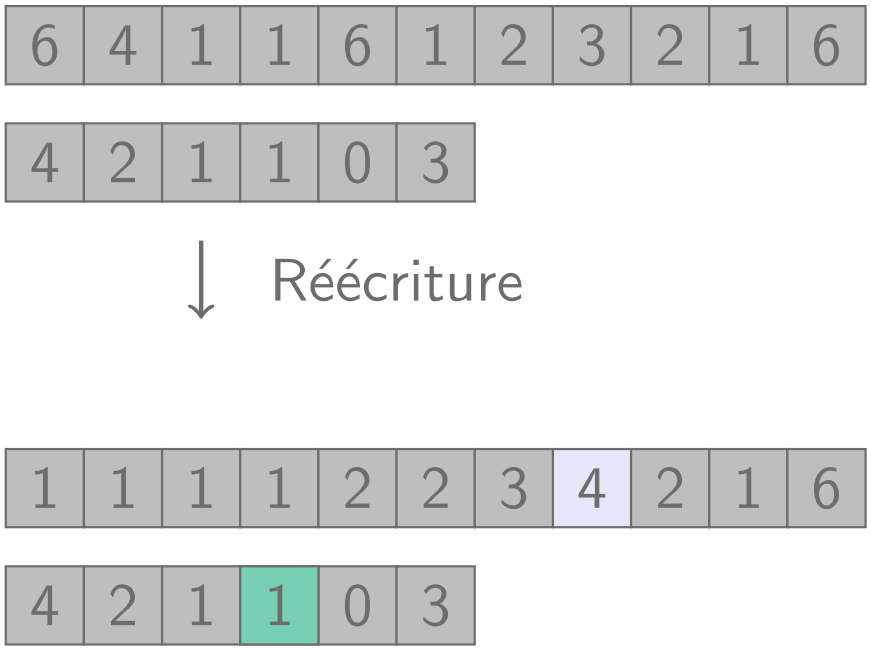
Tri comptage



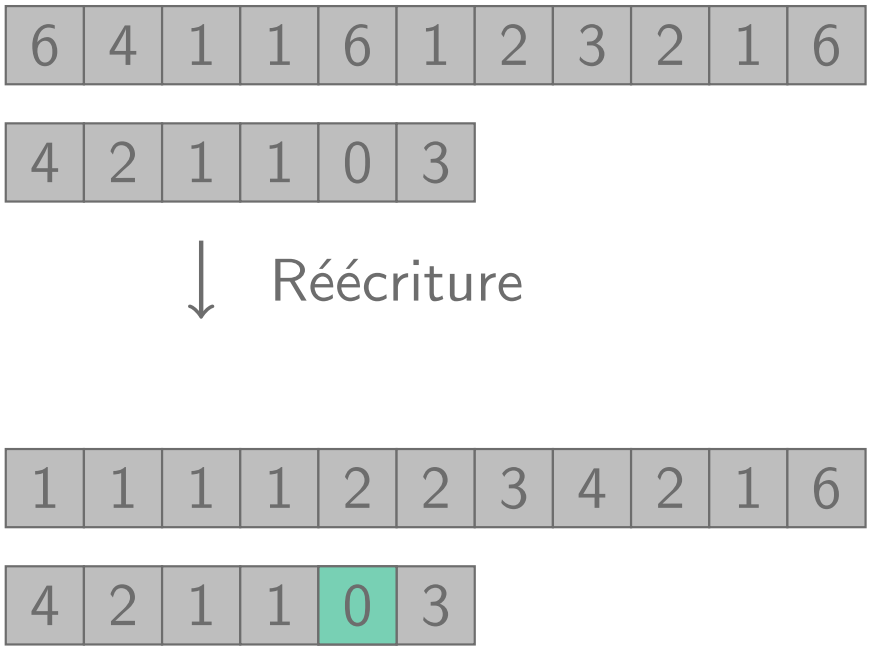
Tri comptage



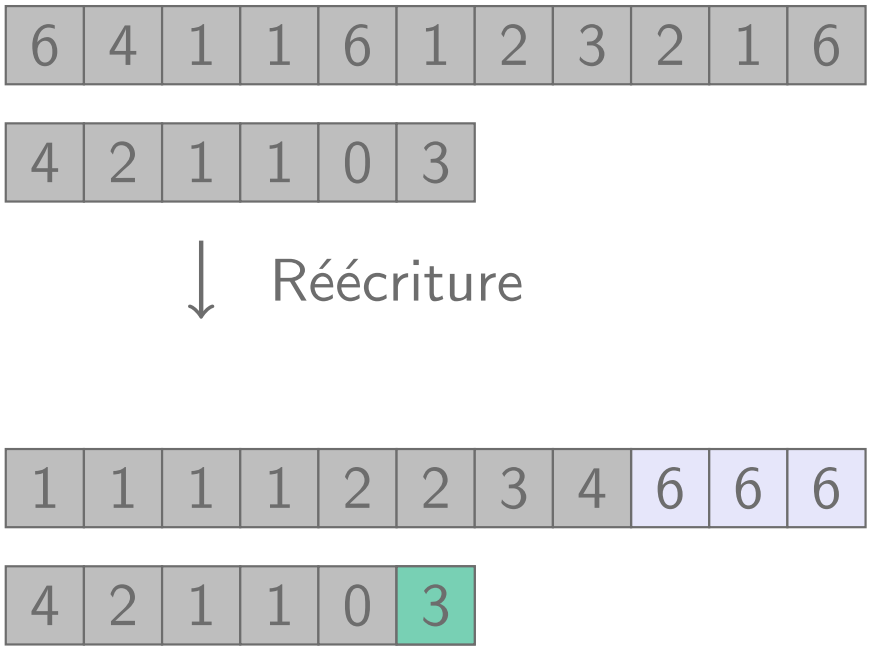
Tri comptage



Tri comptage



Tri comptage



Tri comptage

```
1  ALGORITHME TriComptage(T):
2  VARIABLES:
3     i,j: entiers
4     C : tableau de taille k
5           initialise a 0
6  DEBUT
7     i ← 1
8     TQ i ≤ n FAIRE
9        C[T[i]] ← C[T[i]]+1
10       i ← i+1
11    FTQ
12    j ← 1
13    TQ j ≤ k FAIRE
14       i ← 1
15       TQ i ≤ T[j] FAIRE
16          T[i] = j
17          i ← i+1
18       FTQ
19       j ← j+1
20    FTQ
21 FIN
```

- Complexité en temps :

$$C_{\text{Compt}}(n,k) = \Theta(n+k)$$

- Complexité en espace :

$$E_{\text{Compt}}(n,k) = \Theta(k)$$