« Tripatouille »

Philippe Lac
(philippe.lac@ac-clermont.fr)
Malika More
(malika.more@u-clermont1.fr)

IREM Clermont-Ferrand

Stage Algorithmique

Année 2010-2011

Qu'est-ce-que trier ?

Prototype

- Entrée : un tableau de nombres

6	3	7	2	3	5

- Sortie : le même tableau contenant les mêmes nombres, mais rangés dans l'ordre croissant

2	3	3	5	6	7

Qu'est-ce-que trier ?

Concrètement

Les situations de tri sont très variées

- Trier des mots par ordre alphabétique
- Trier des fichiers par longueur
- Trier des messages par date
- Trier des articles par référence
- Trier des personnes par date de naissance
- etc.

Qu'est-ce-que trier ?

En informatique

- On trie de très grandes quantités de données (quelques centaines de milliers d'éléments) à tout bout de champ.
- C'est pourquoi les informaticiens ont inventé de très nombreux algorithmes de tri, plus ou moins rapides et efficaces.
- Les ordinateurs passent énormément de temps à faire des tris : on considère aujourd'hui que les algorithmes de tri sont ceux qui sont les plus utilisés par les ordinateurs du monde entier !

Pourquoi parler des tris ?

Sujet incontournable en algorithmique :

- Premiers algorithmes vraiment utiles dans un cursus d'algorithmique (e.g. en DUT Informatique)
- Culture générale : utilisés de façon visible ou non pour l'utilisateur un peu partout en informatique (en particulier en programmation système, en bases de données, en programmation web, etc.)
- Cas d'école en algorithmique : plusieurs algorithmes très différents pour le même problème
- Occasion idéale pour parler de l'efficacité des algorithmes, c'est-à-dire de *complexité algorithmique*

Présentation

On le trouve à l'adresse :

`http://www.malgouyres.fr/tripatouille/`

Il s'agit d'une application permettant :

- De simuler à la main des algorithmes de tri
- De voir des démonstrations pas à pas des trois algorithmes de tri quadratiques classiques (tri par sélection, tri par insertion, tri par bulles).

On peut l'utiliser dans un navigateur internet à l'adresse indiquée ou la télécharger à la même adresse.

Démonstration

Exercice

1. Dans l'application `Tripatouille` cliquer sur `Nouveau Tableau`, puis aller dans l'onglet `Trier à la main` et cliquer sur `Tout cacher`.
2. Trier ce tableau. Les seules actions possibles sont les suivantes :
   - Cliquer sur une case cachée la rend visible et cliquer sur une case visible la cache (deux cases écrites en noir au maximum peuvent être visibles en même temps).
   - Lorsque deux cases écrites en noir exactement sont visibles, on peut échanger leur contenu en cliquant sur le bouton `Échanger`.
3. Proposer un algorithme de tri.

Commentaires

- Les problèmes de tri ne présentant aucune difficulté mathématique, leur étude en introduction à l'algorithmique permet d'aborder en douceur des notions importantes et spécifiques (instructions, mémoire, instructions de contrôle).
- Par contre, à ce stade, il me semble inutile d'imposer un formalisme particulier pour rédiger les algorithmes, au risque de noyer les notions sous la technique
- On pourrait évidemment faire faire les mêmes exercices avec des cartes portant des nombres sur une seule face, l'ordinateur n'est qu'un cosmétique.

Notions importantes

- Les **instructions** de traitement des données sont **limitées et stéréotypées** : dans cet exemple, la seule possibilité est d'échanger le contenu de deux cases du tableau
- La **mémoire** est représentée par des variables en nombre fixé à l'avance : par conséquent, à chaque instant on n'a qu'une vision **partielle et locale** du tableau, qui correspond ici aux deux cases visibles au maximum
- Des **instructions de contrôle** (alternatives et boucles) sont nécessaires à la description d'un algorithme, puisqu'il doit fonctionner pour un tableau de nombres quelconques de longueur quelconque

Traiter des séries de données

Exemple (Un triangle $M_1M_2M_3$ est-il isocèle en M_1 ?)

- Soient Les points

$$M_1(x_1, y_1) \qquad M_2(x_2, y_2) \qquad M_3(x_3, y_3)$$

- On compare les longueurs

$$M_1M_2 \qquad \text{et} \qquad M_1M_3$$

- où

$$M_1M_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$M_1M_3 = \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2}$$

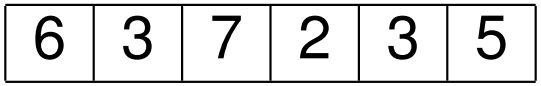
Traiter des séries de données

Exemple (Un triangle $M_1M_2M_3$ est-il isocèle en M_1 ?)

Algorithme 2 : TriangleIsocèle

début
  pour *i* **de** *1* **à** *3* **faire**
    Lire la valeur de $abs[i]$ `% série des abscisses %`
    Lire la valeur de $ord[i]$ `% série des ordonnées %`
  fin
  Donner à $d1$ la valeur $\sqrt{(abs[2]-abs[1])^2+(ord[2]-ord[1])^2}$
  Donner à $d2$ la valeur $\sqrt{(abs[3]-abs[1])^2+(ord[3]-ord[1])^2}$
  si $d1 = d2$ **alors**
    Afficher "C'est un triangle isocèle en M_1"
  sinon
    Afficher "Ce n'est pas un triangle isocèle en M_1"
  fin
fin

Notions élémentaires



Tableau

- Utilisé pour stocker plusieurs données de même type
- Type des données : simple ou élaborée
- Longueur du tableau : nombre de cases n
- Indices des cases : de 1 à n ou de 0 à $n-1$ ou autre
- Accès direct au contenu de la case d'indice i par $T[i]$
- Permet un traitement séquentiel par des boucles

Exemple d'utilisation

Exercice

Écrire un algorithme prend en entrée une valeur n et renvoie la moyenne de n nombres pris au hasard entre 1 et 100

Exemple d'utilisation

Solution

Fonction `Moyenne`(*n : entier*)

variables locales : *i*, *somme* : entiers, *moyenne* : flottant, *T* : tableau de *n* entiers

début

  pour *i* **de** *1* **à** *n* **faire**

    Donner à $T[i]$ une valeur au hasard entre 1 et 100

  fin

  Donner à *somme* la valeur 0

  pour *i* **de** *1* **à** *n* **faire**

    Donner à *somme* la valeur $somme + T[i]$

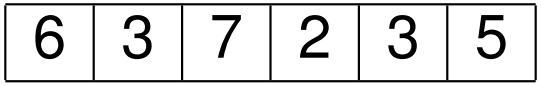
  fin

  Donner à *moyenne* la valeur $somme/n$

  retourner : *moyenne*

fin

Notions avancées



Tableau

- Tableau à plusieurs dimensions $\longrightarrow$ matrices etc.
- Si le nombre d'éléments peut varier en cours de calcul, on préfèrera souvent utiliser une liste (à suivre...)
- Dans les langages de programmation, les indices des cases commencent obligatoirement à une valeur fixée (en général 0 ou 1)
- En termes de complexité, le nombre d'éléments est en général une donnée plus pertinente que le type des éléments

Algorithmes de tri

Deux grandes familles :

Algorithmes simples

- Tri par sélection
- Tri par insertion
- Tri à bulles

Algorithmes élaborés

- Tri rapide
- Tri fusion
- Tri par tas
- Etc.

Tris « en place »

Convention

- Les nombres sont réarrangés à l'intérieur du tableau donné en entrée
- À chaque instant, seulement un nombre constant d'entre eux est stocké à l'extérieur du tableau
- Quand l'algorithme s'arrête, le résultat est contenu dans le même tableau qu'au départ

Algodiversité

- Chacun de ces algorithmes possède des champs d'application privilégiés.
- Par exemple, le tri par sélection et le tri par insertion sont très simples et efficaces lorsqu'on doit trier un petit nombre d'éléments (jusqu'à 10 environ).
- Pour des données plus nombreuses, le tri rapide est le plus utilisé actuellement. Mais s'il est rapide en moyenne, il est par contre très lent lorsque le tableau d'entrée est déjà presque trié.
- Si on sait à l'avance que c'est le cas, on préfèrera utiliser le tri par tas.
- Il existe aussi de nombreux algorithmes de tri plus spécialisés, comme le tri par base (radix sort).

Tri par sélection

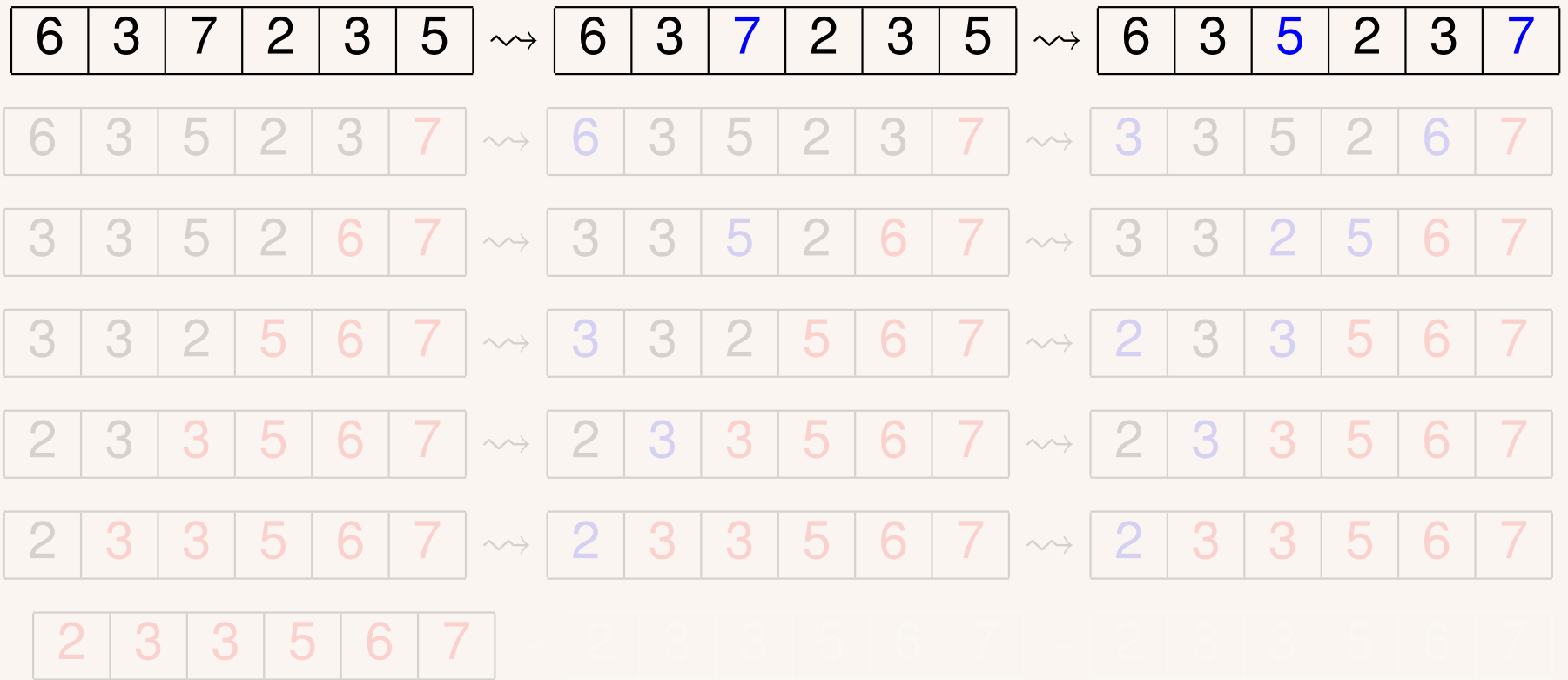
6	3	7	2	3	5

Idée

- Sélectionner le plus grand élément du tableau
- Le positionner en dernière position par un échange
- Recommencer sur le reste du tableau

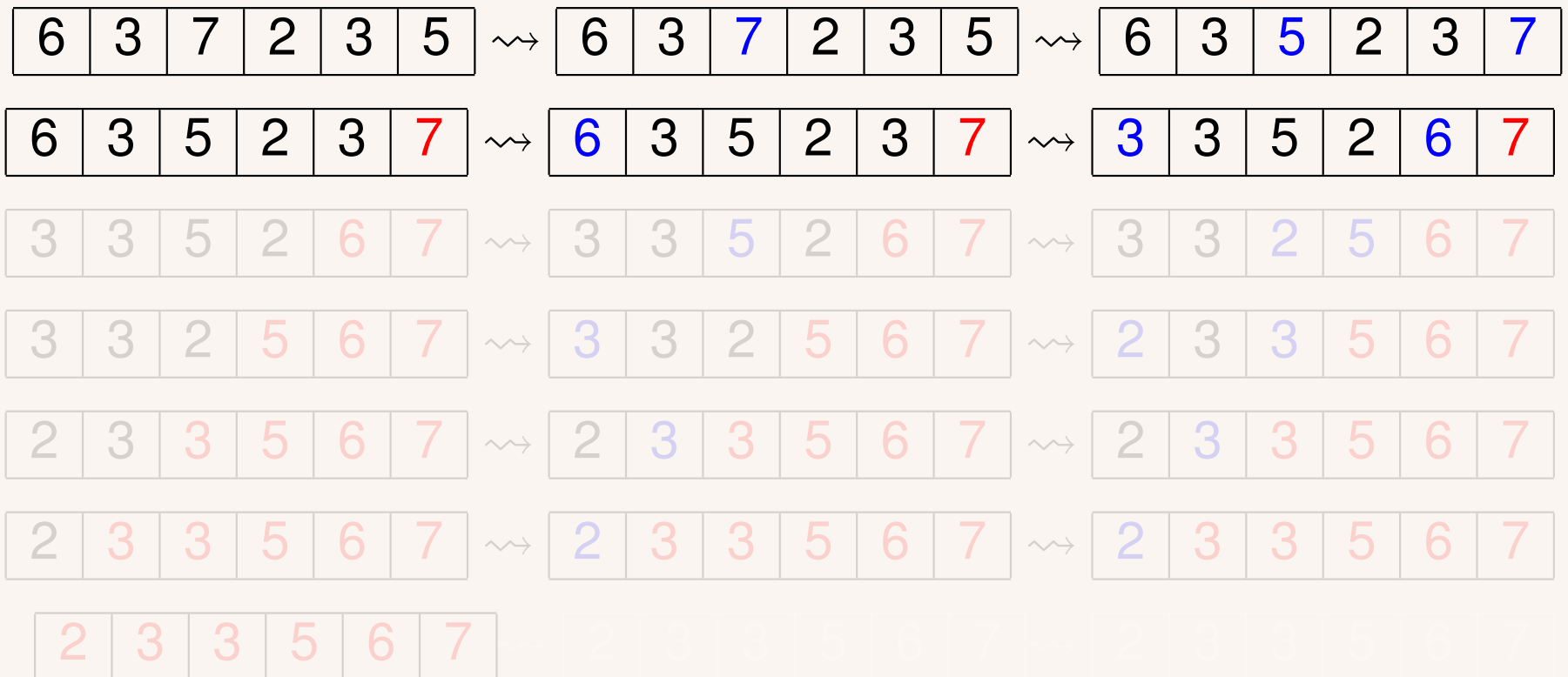
Tri par sélection

Exemple



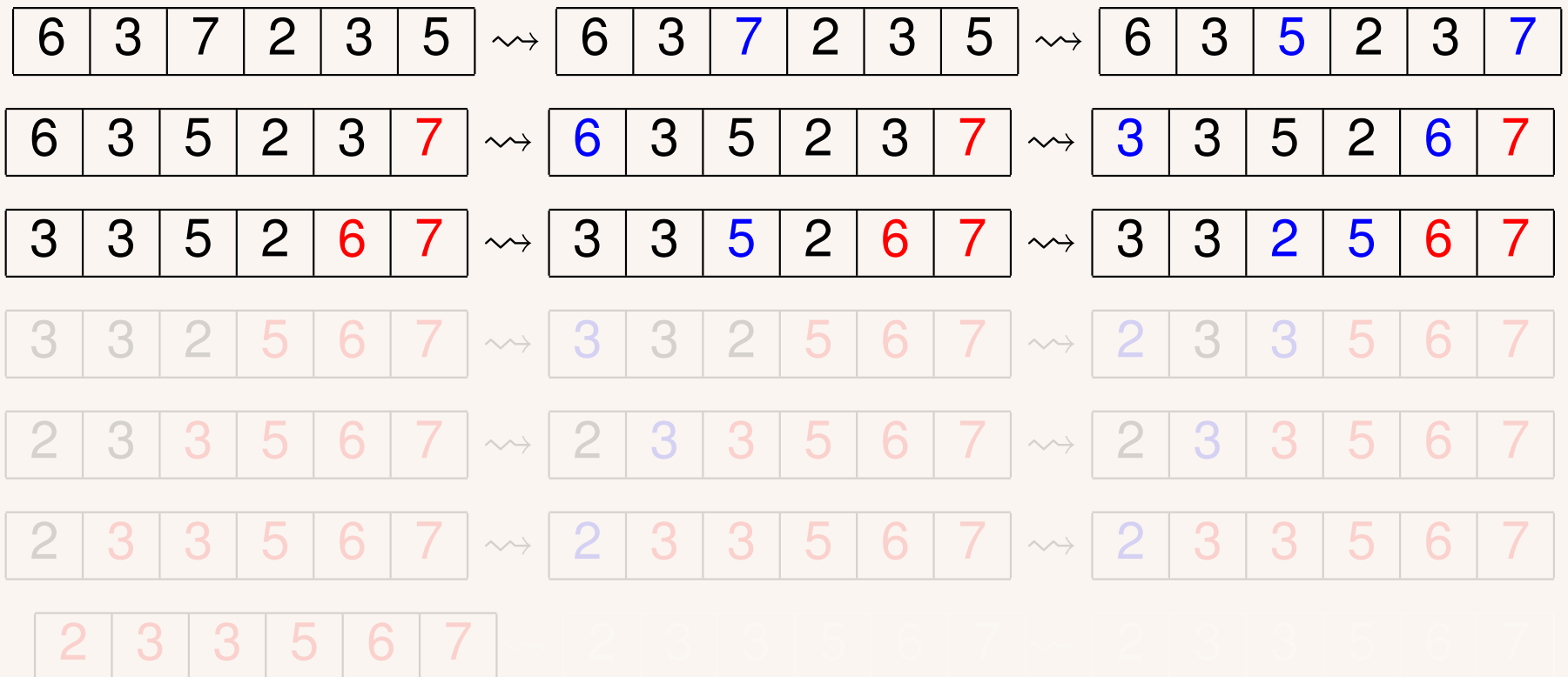
Tri par sélection

Exemple



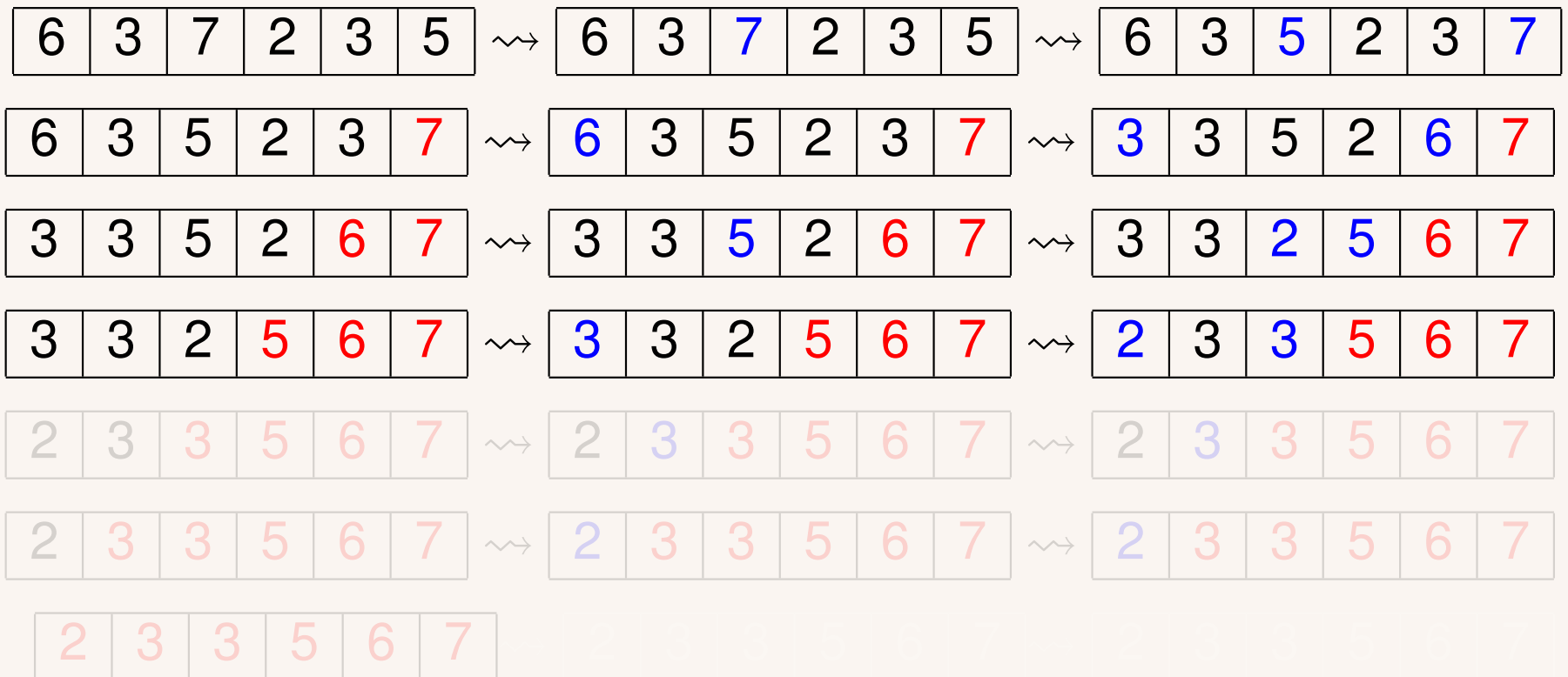
Tri par sélection

Exemple



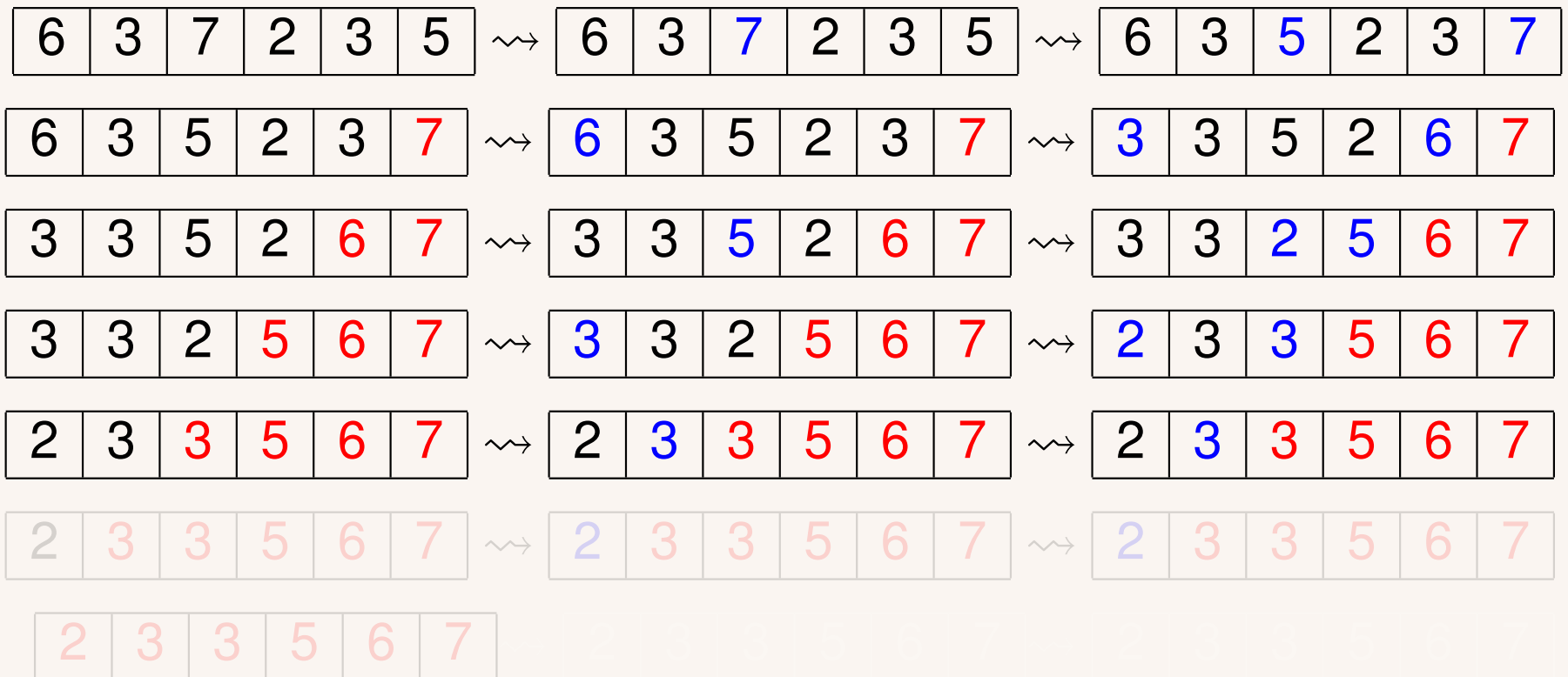
Tri par sélection

Exemple



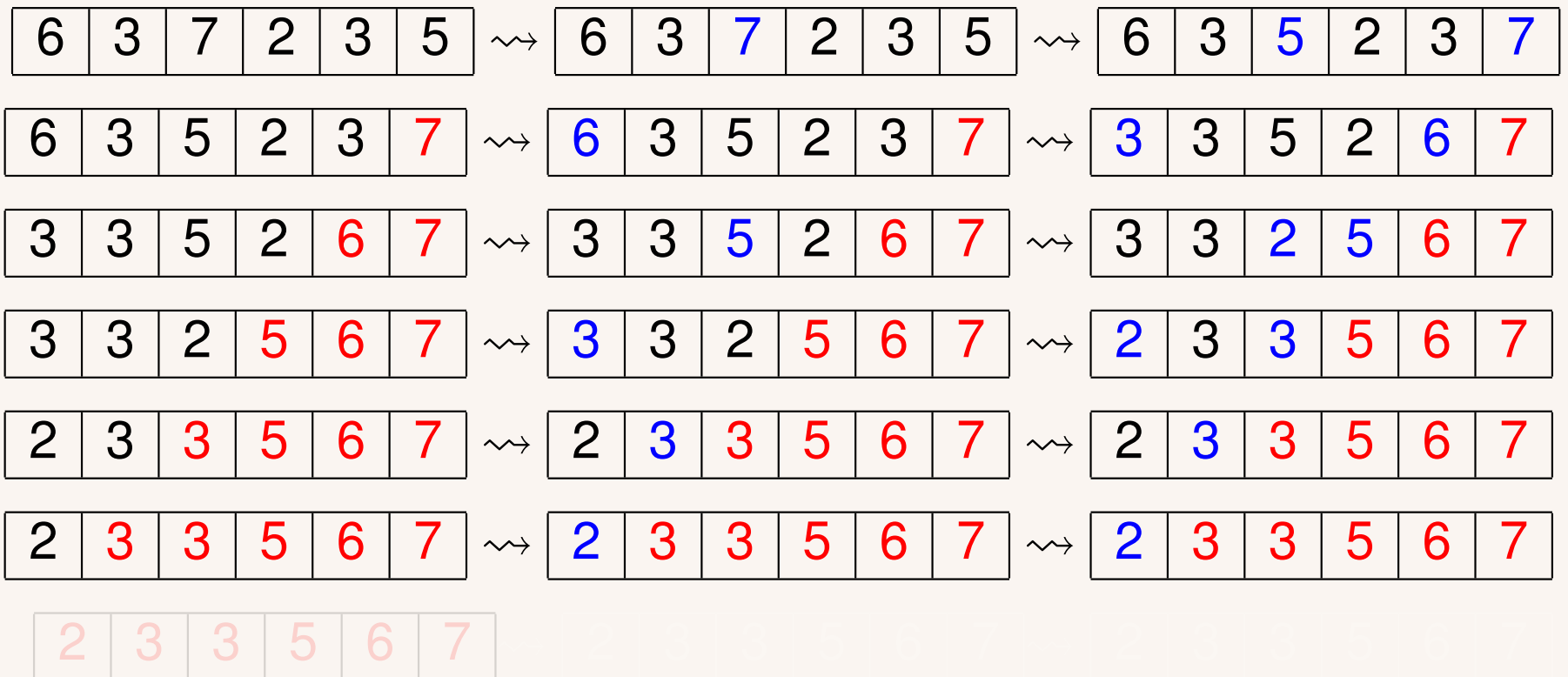
Tri par sélection

Exemple



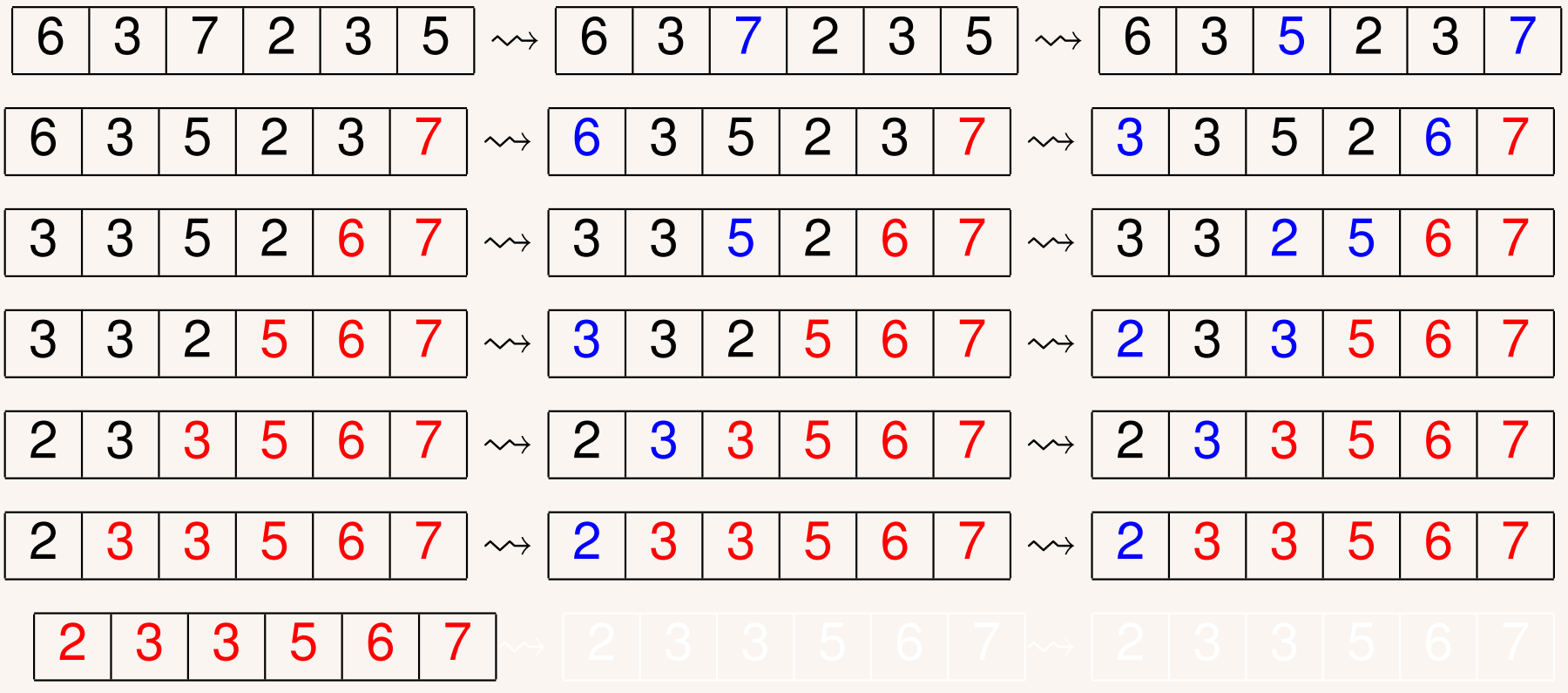
Tri par sélection

Exemple



Tri par sélection

Exemple



Algorithme 5 : Tri par sélection

Entrée : Un tableau T de n entiers
Résultat : Le tableau T trié

```
début
    variables locales : Des entiers k, i, imax, temp
    pour k de n à 2 par pas de −1 faire
        % recherche de l'indice du maximum : %
        Donner à imax la valeur 1
        pour i de 2 à k par pas de 1 faire
            si T[imax] < T[i] alors
                Donner à imax la valeur i
            fin
        fin
        % échange : %
        Donner à temp la valeur T[k]
        Donner à T[k] la valeur T[imax]
        Donner à T[imax] la valeur temp
    fin
fin
```

Tri par insertion

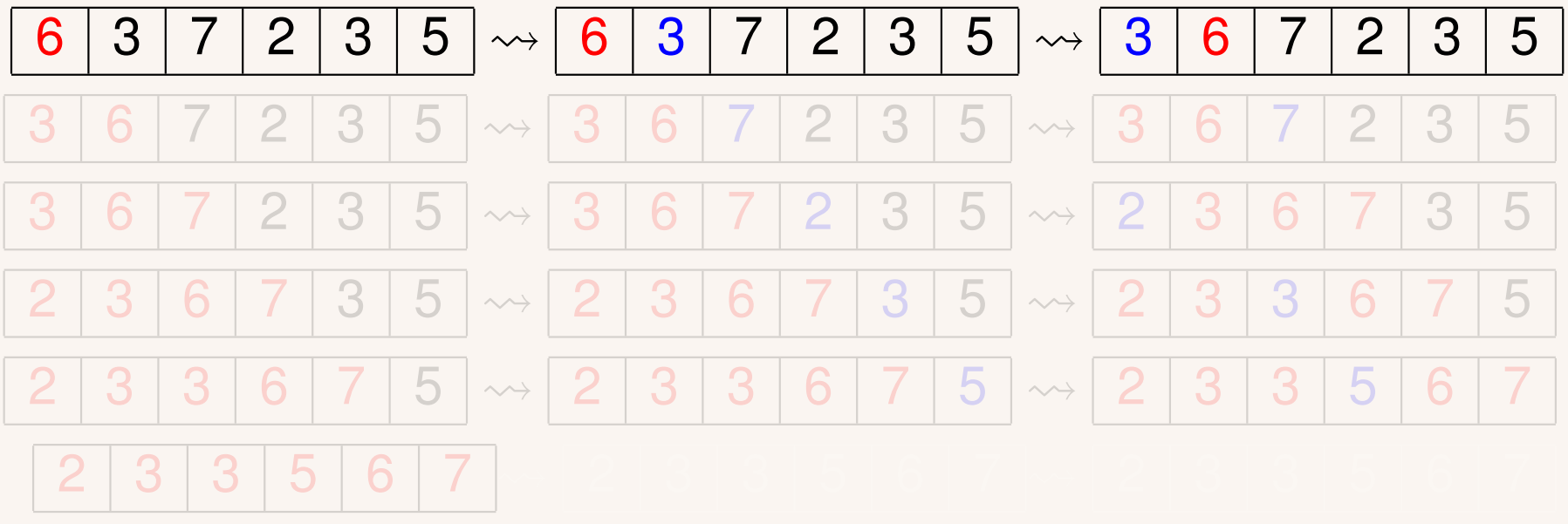
6	3	7	2	3	5

Idée

- Le premier élément du tableau est déjà trié
- Insérer le deuxième élément à sa place pour obtenir une liste triée à deux éléments
- Insérer le troisième élément à sa place pour obtenir une liste triée à trois éléments
- Et ainsi de suite

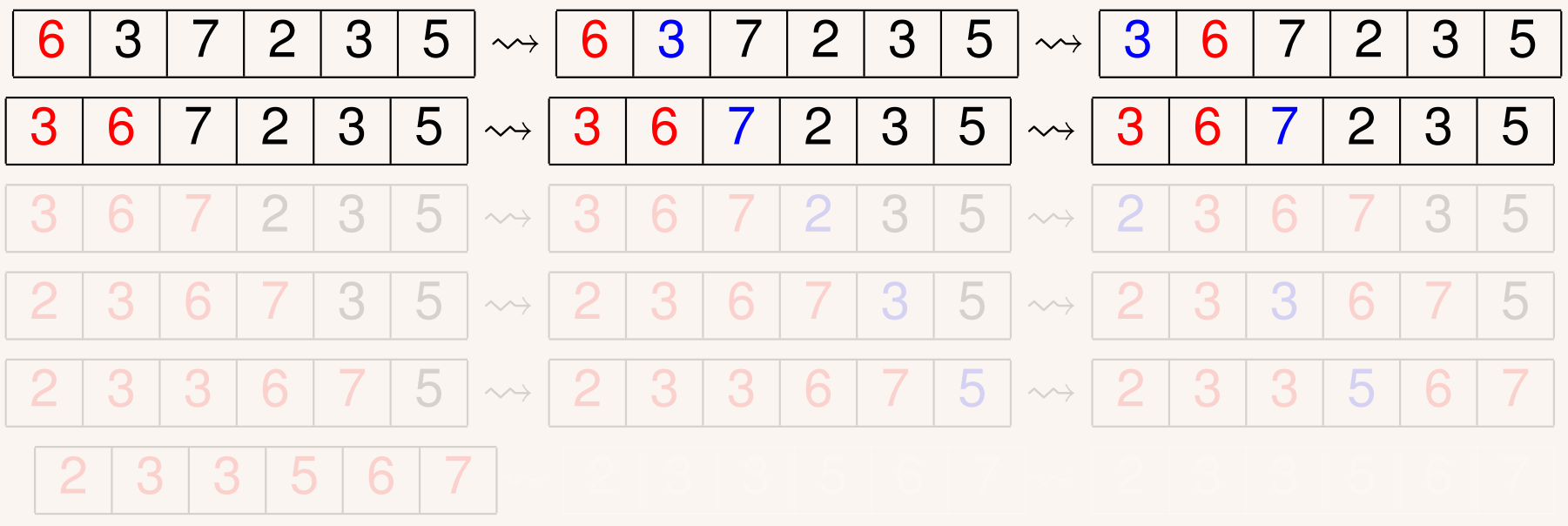
Tri par insertion

Exemple



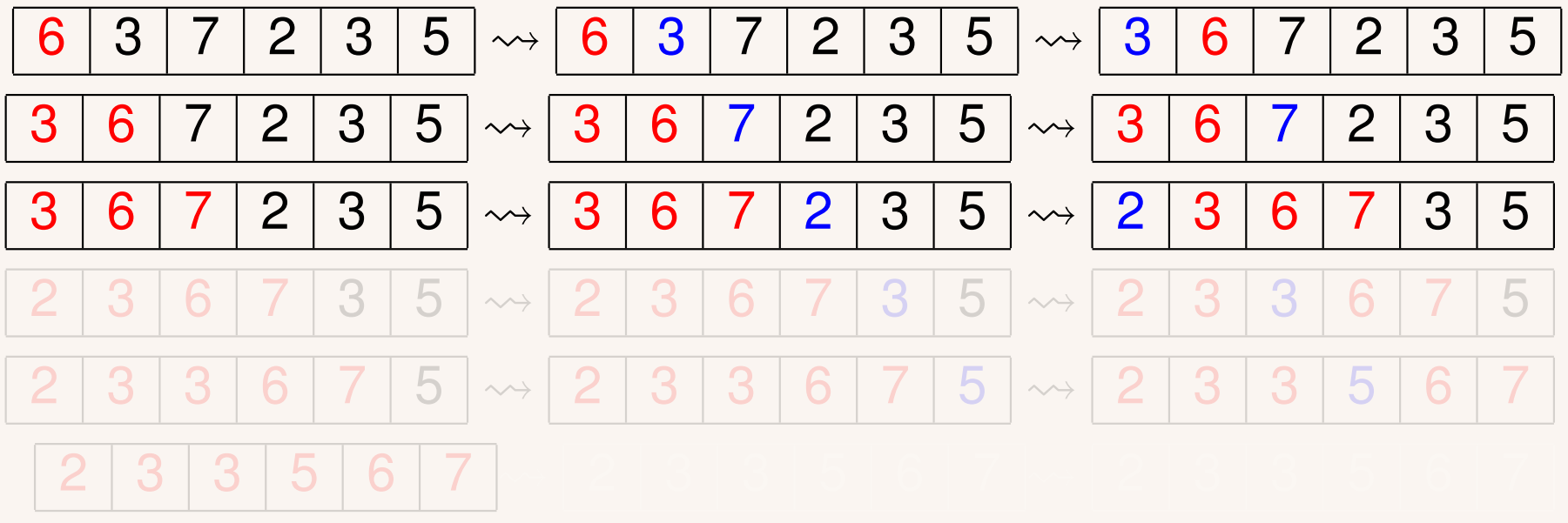
Tri par insertion

Exemple



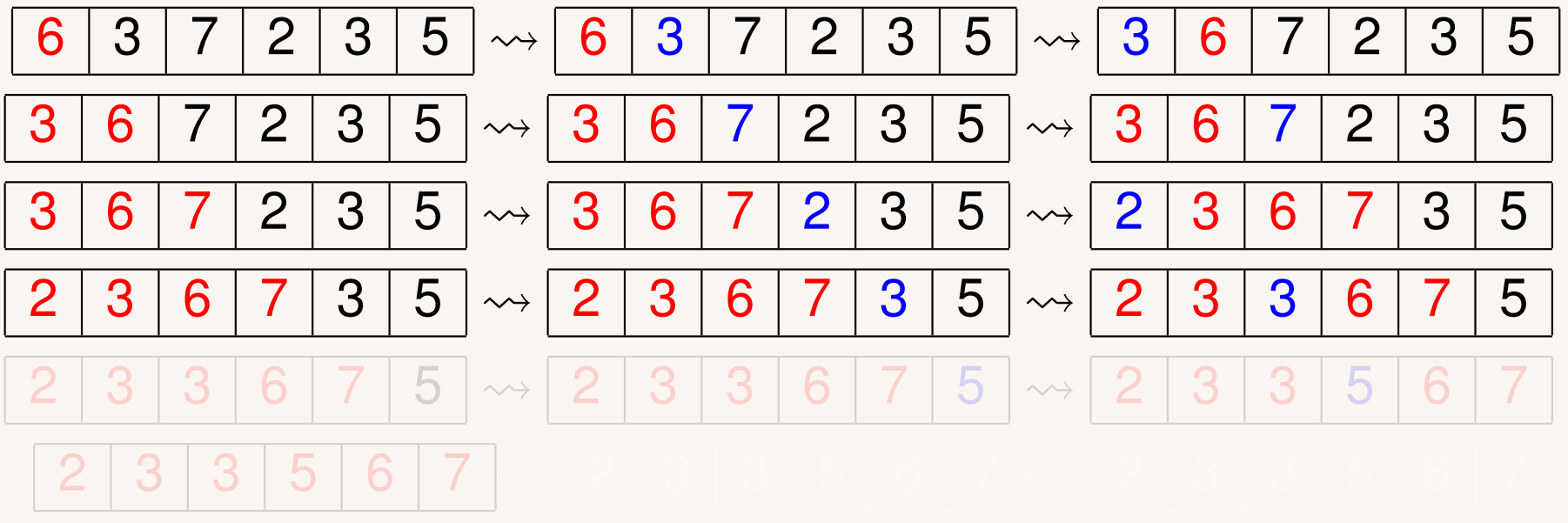
Tri par insertion

Exemple



Tri par insertion

Exemple

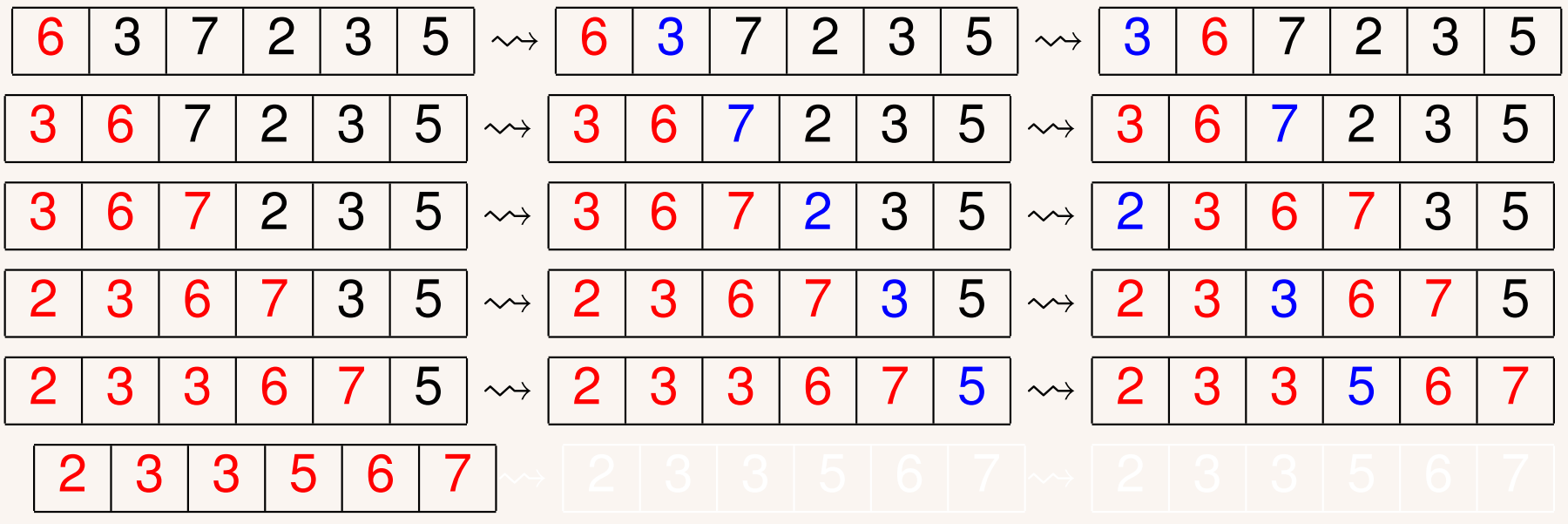


Tri par insertion

Exemple

Tri par insertion

Exemple



Algorithme 6 : Tri par insertion

Entrée : Un tableau T de n entiers
Résultat : Le tableau T trié

début
  variables locales : Des entiers k, i, v
  pour k **de** 2 **à** n **par pas de** 1 **faire**
    Donner à v la valeur $T[k]$
    Donner à i la valeur $k-1$
    % décalage des éléments pour l'insertion : %
    tant que $i \geq 1$ **et** $v < T[i]$ **faire**
      Donner à $T[i+1]$ la valeur $T[i]$
      Donner à i la valeur $i-1$
    fin
    % insertion proprement dite : %
    Donner à $T[i+1]$ la valeur v
  fin
fin

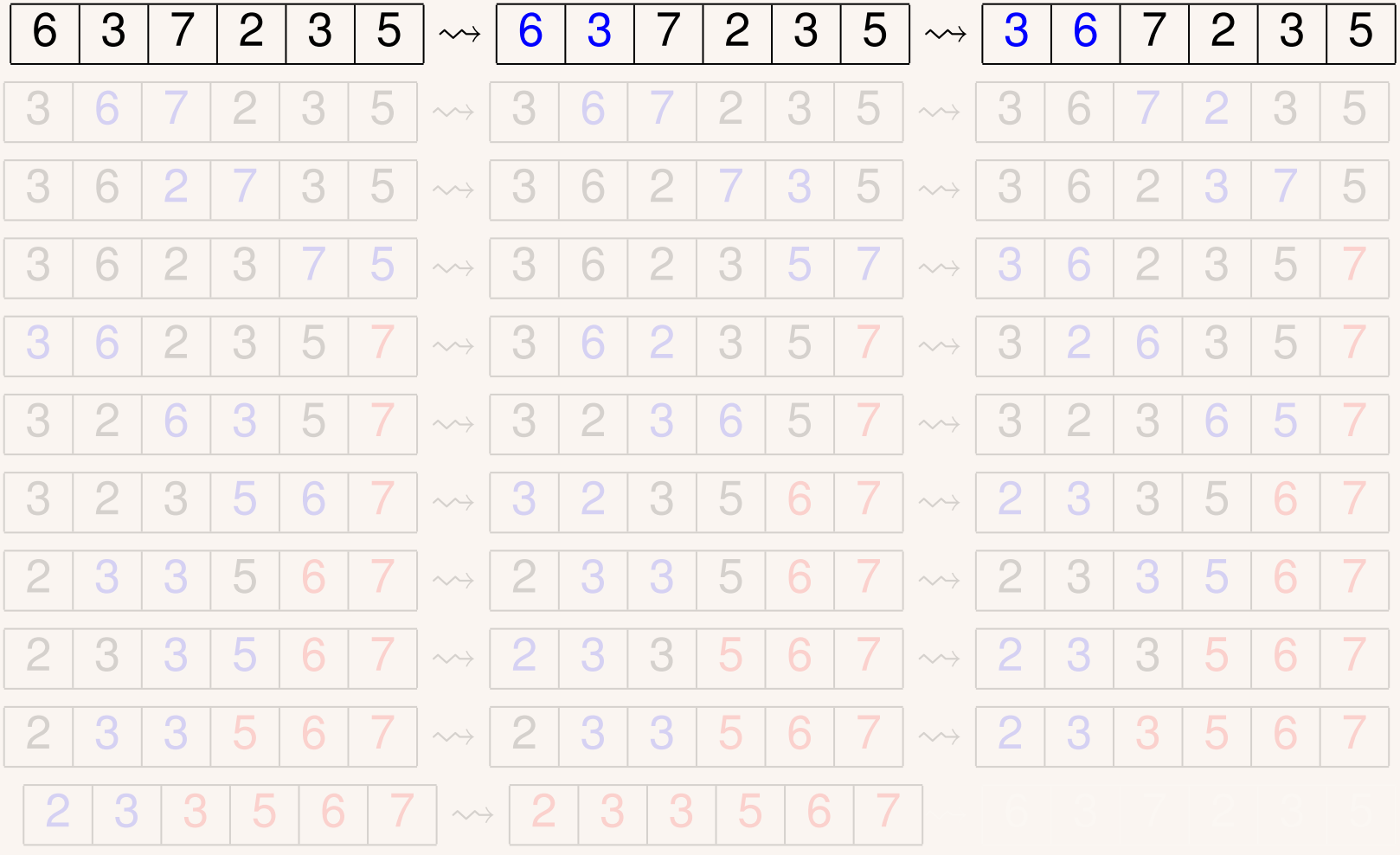
Tri par bulles

Idée

- Échanger les deux premiers éléments s'ils ne sont pas dans l'ordre
- Échanger le deuxième et le troisième éléments s'ils ne sont pas dans l'ordre
- Ainsi de suite
- La première passe amène le maximum à sa place définitive
- La deuxième passe amène l'élément juste inférieur au maximum à sa place définitive
- Ainsi de suite

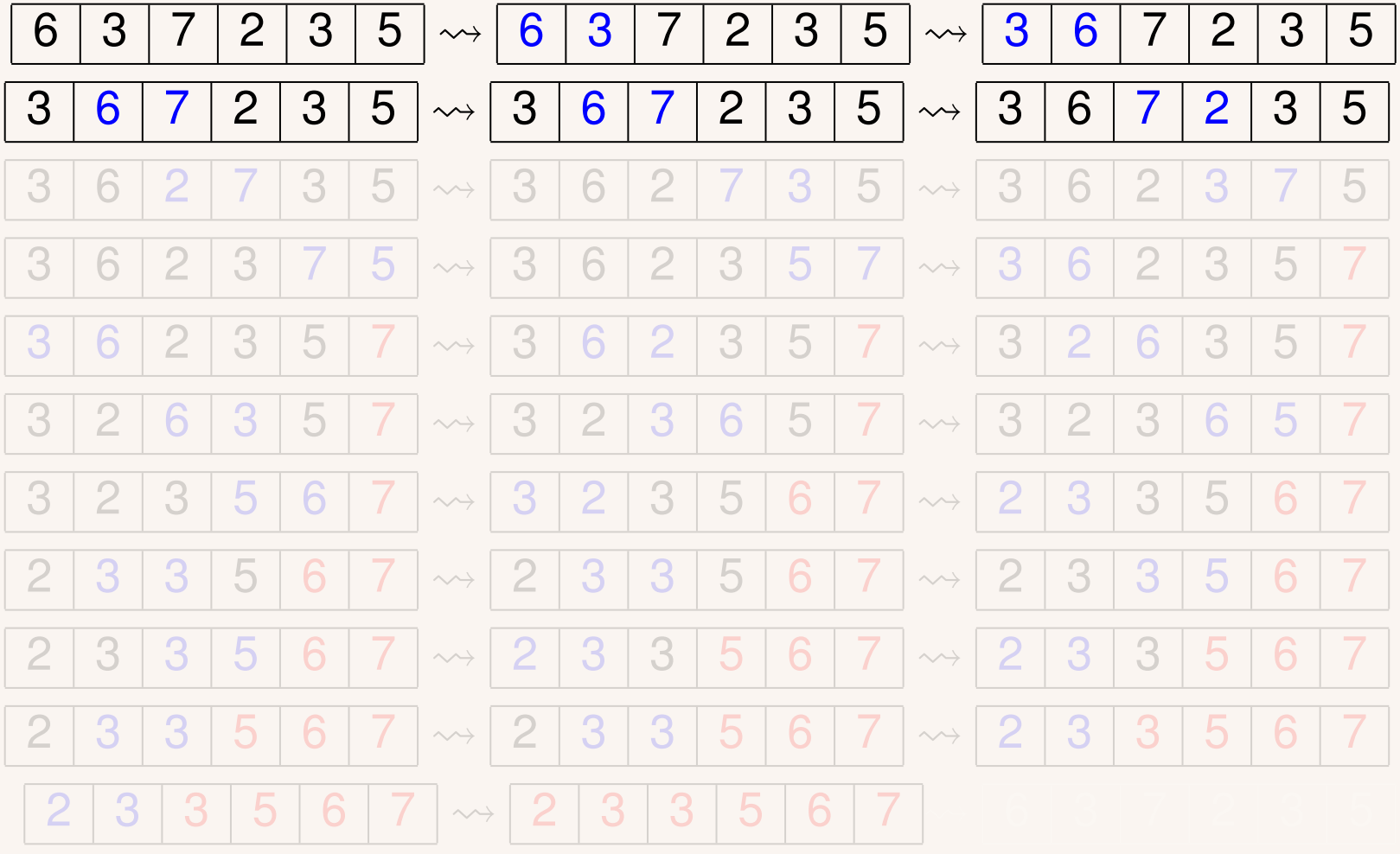
Tri par bulles

Exemple



Tri par bulles

Exemple



Tri par bulles

Exemple

Tri par bulles

Exemple

Tri par bulles

Exemple

Tri par bulles

Exemple

Tri par bulles

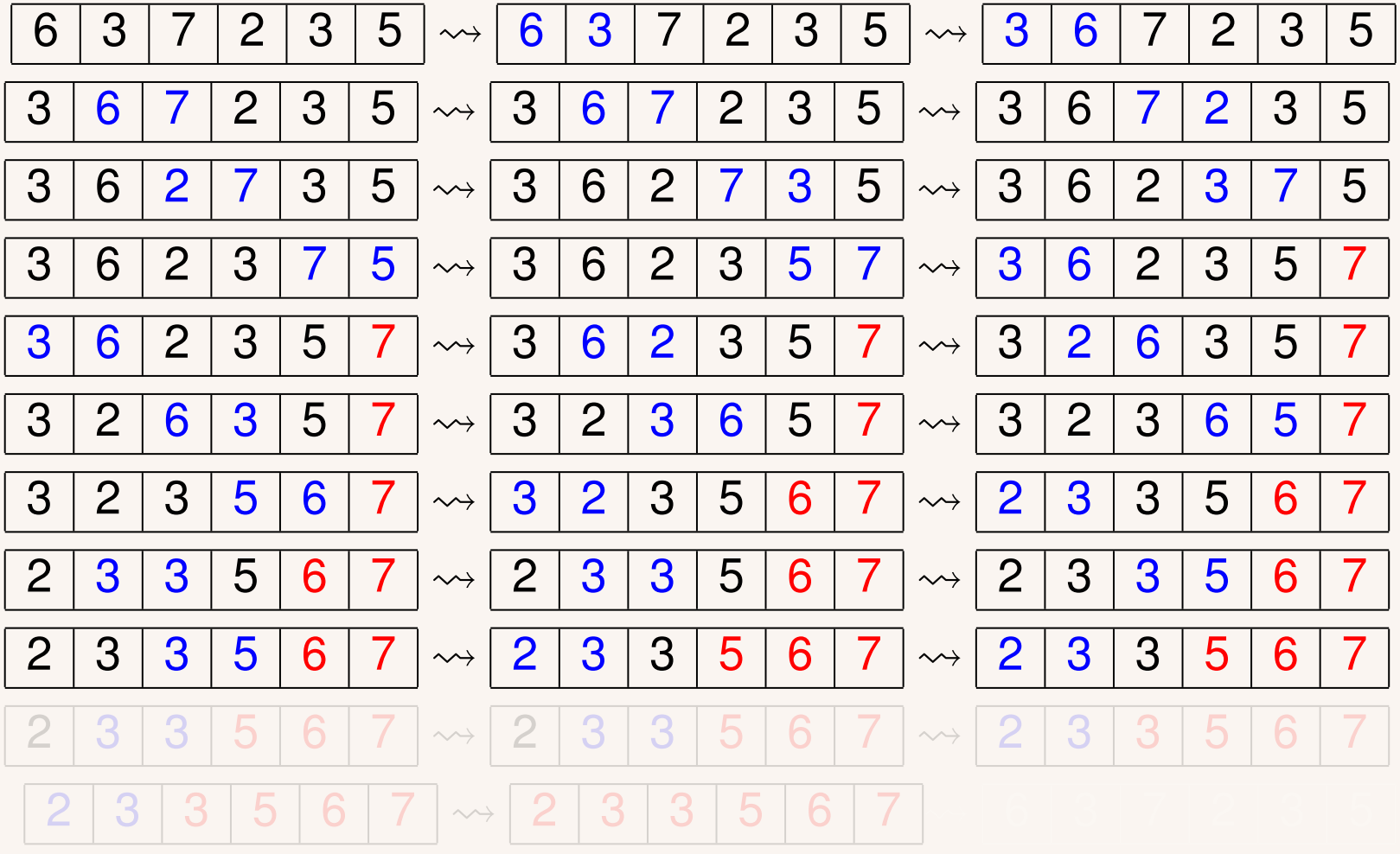
Exemple

Tri par bulles

Exemple

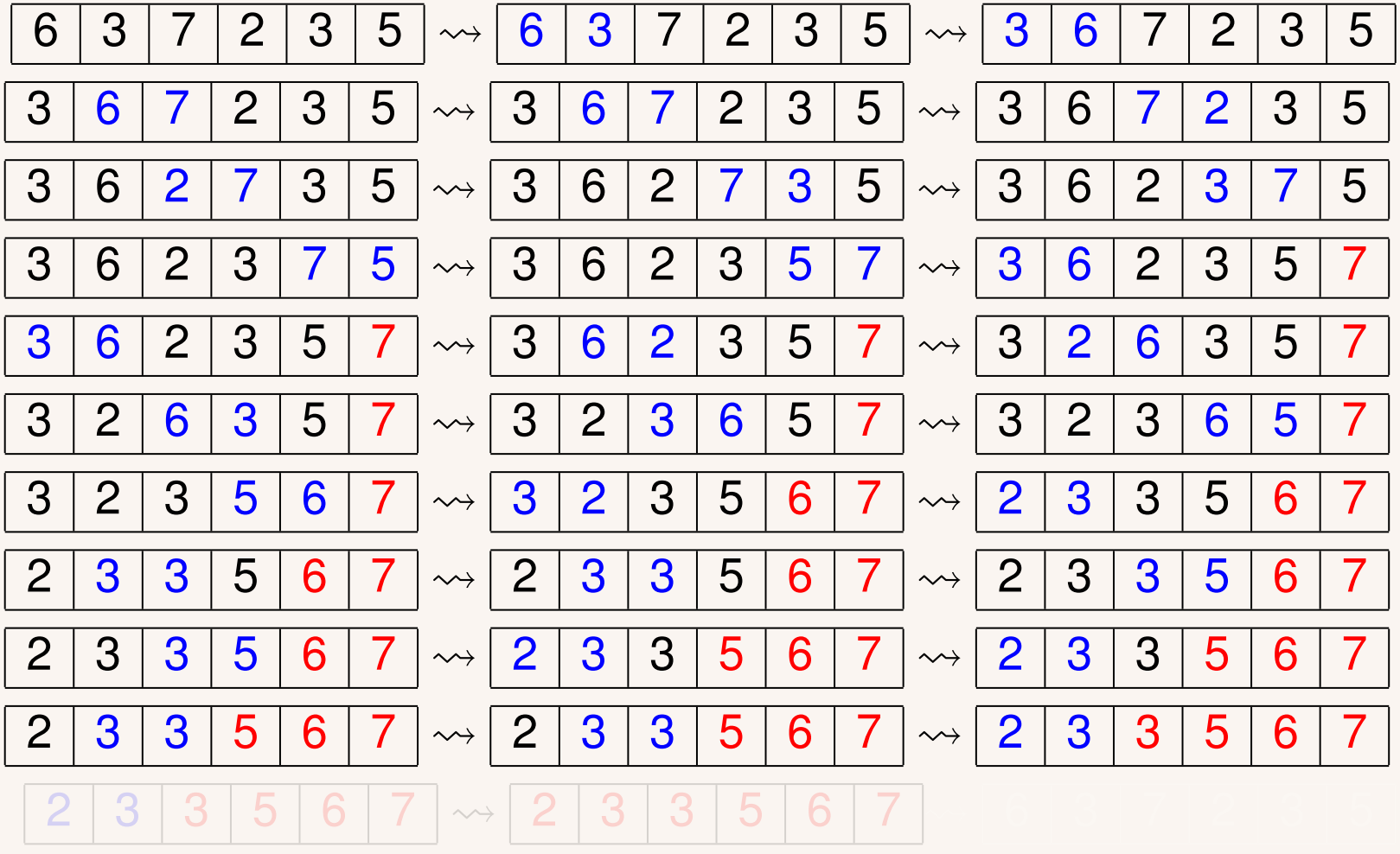
Tri par bulles

Exemple



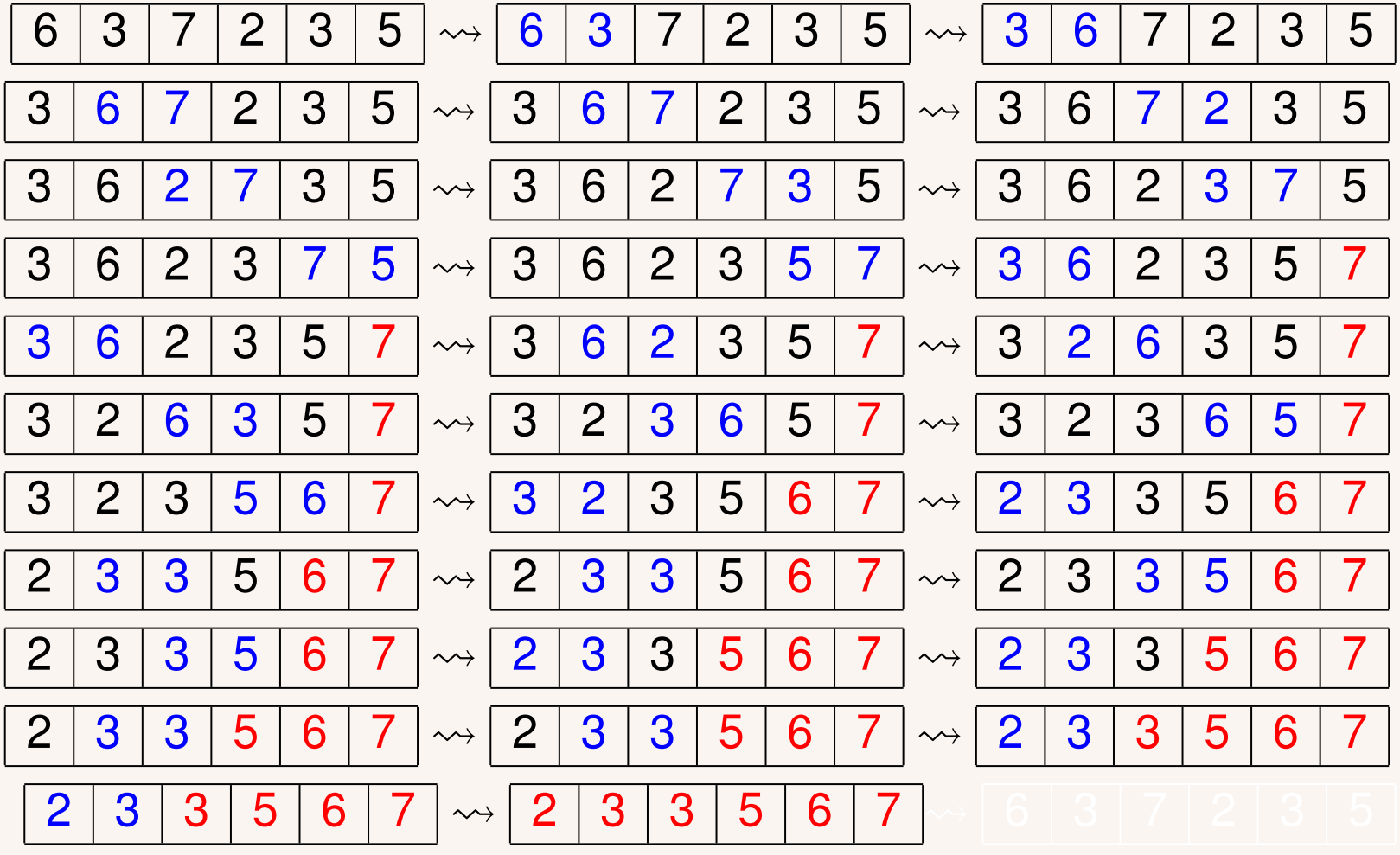
Tri par bulles

Exemple



Tri par bulles

Exemple



Algorithme 7 : Tri par bulles

Entrée : Un tableau T de n entiers
Résultat : Le tableau T trié

```
début
    variables locales : Des entiers k, i, temp
    % pour chaque passe : %
    pour k de n à 2 par pas de −1 faire
        % on fait remonter le plus grand : %
        pour i de 2 à k par pas de 1 faire
            si T[i] < T[i − 1] alors
                % échange de T[i] et de T[i − 1] : %
                Donner à temp la valeur T[i]
                Donner à T[i] la valeur T[i − 1]
                Donner à T[i − 1] la valeur temp
            fin
        fin
    fin
fin
```

FIN