1M - Informatique
Cours 2 - Algorithmique

A. Ridard

A propos de ce document

- Pour naviguer dans le document, vous pouvez utiliser :
  - le menu (en haut à gauche)
  - l'icône en dessous du logo GyYv
  - les différents liens
- Pour signaler une erreur, vous pouvez envoyer un message à l'adresse suivante : anthony.ridard@eduvaud.ch

Ne pas confondre algorithme et programme

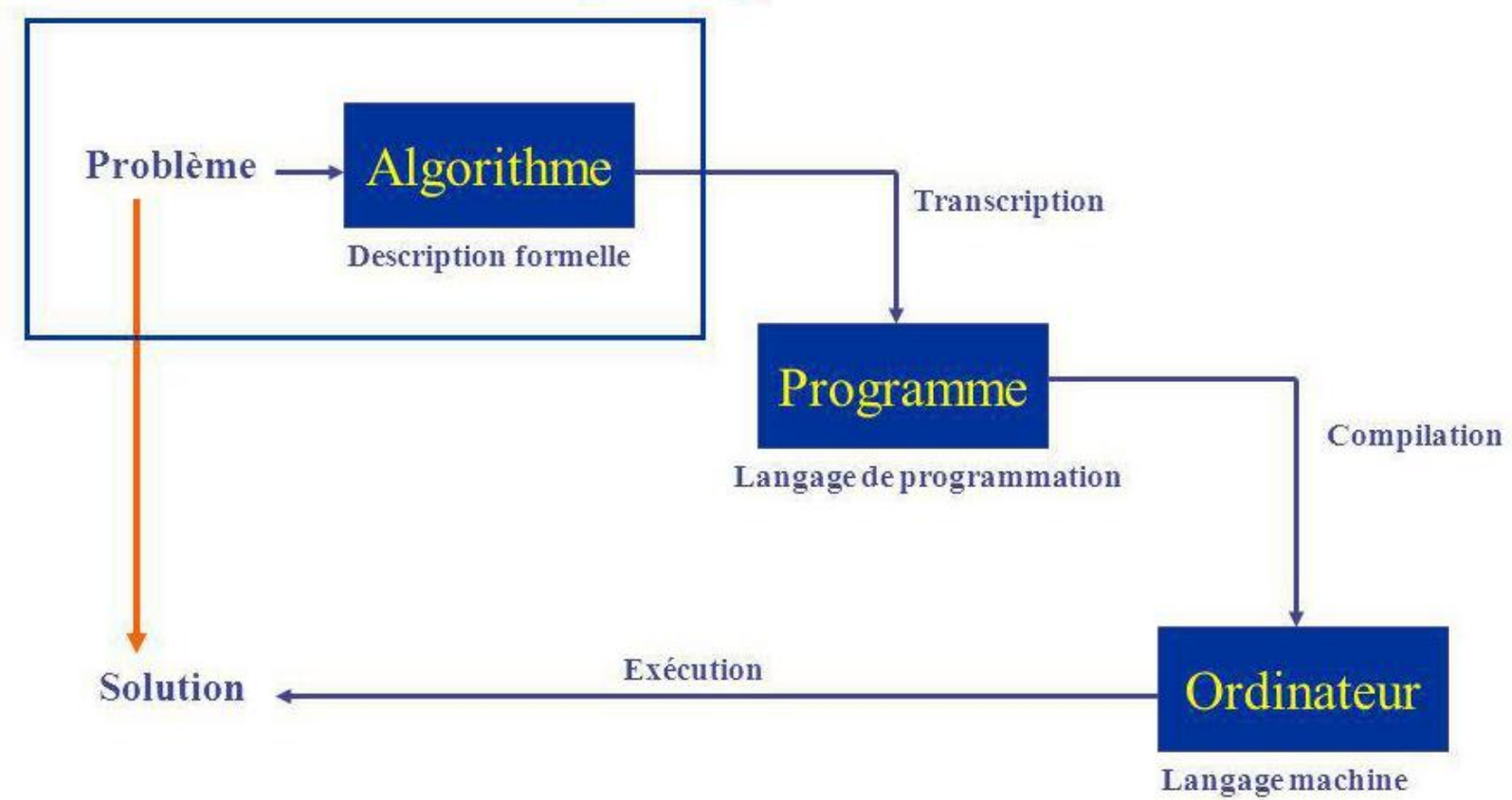
Pour résoudre un problème à l'aide d'un ordinateur, on suit le processus suivant :

1. On conçoit un algorithme c'est à dire une méthode de résolution, disons une « recette » décrite dans un langage « humain » [a] en précisant bien :
   - les ingrédients utilisés (données en entrée)
   - les différentes étapes à réaliser (dans un certain ordre)
   - le résultat obtenu
2. On transforme cet algorithme en un programme écrit dans un langage de programmation (Python, C, ...)
3. L'interpréteur (Python) ou le compilateur (C) transforme ce programme dans un langage machine (une suite de 0 et de 1) exécutable par l'ordinateur qui fournit alors la solution au problème initial

---

a. Ce langage est appelé pseudo-code

Gymnase Yverdon

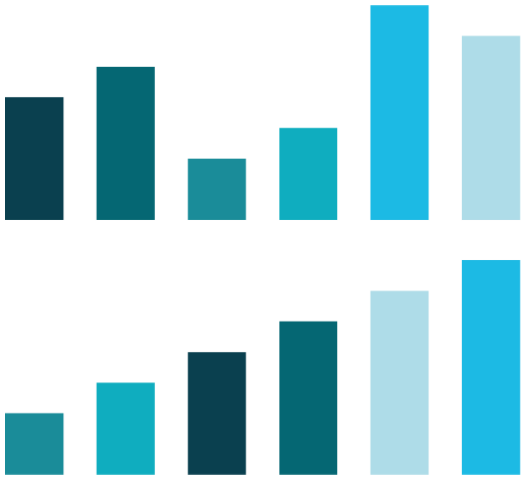

Problème
Algorithme
Description formelle
Transcription
Programme
Langage de programmation
Compilation
Solution
Exécution
Ordinateur
Langage machine

- Lorsque plusieurs algorithmes existent, on choisit le « meilleur » [a]
- Un problème est souvent découpé en sous-problèmes que l'on sait résoudre
- Exemples :
  - Algorithme de reconnaissance faciale (natel)
  - Algorithme de recommandation (posts, amis, films, séries, produits)
  - Algorithme de classification (banques, assurances)
  - Algorithme de détection de fraude (fiscale)
  - Algorithme de recherche (moteur google)
  - Algorithme de tri

---

a. Pour cela, on utilise un critère à optimiser (le temps d'exécution lié au nombre d'opérations élémentaires, l'espace mémoire utilisé, ...)

Problème du tri

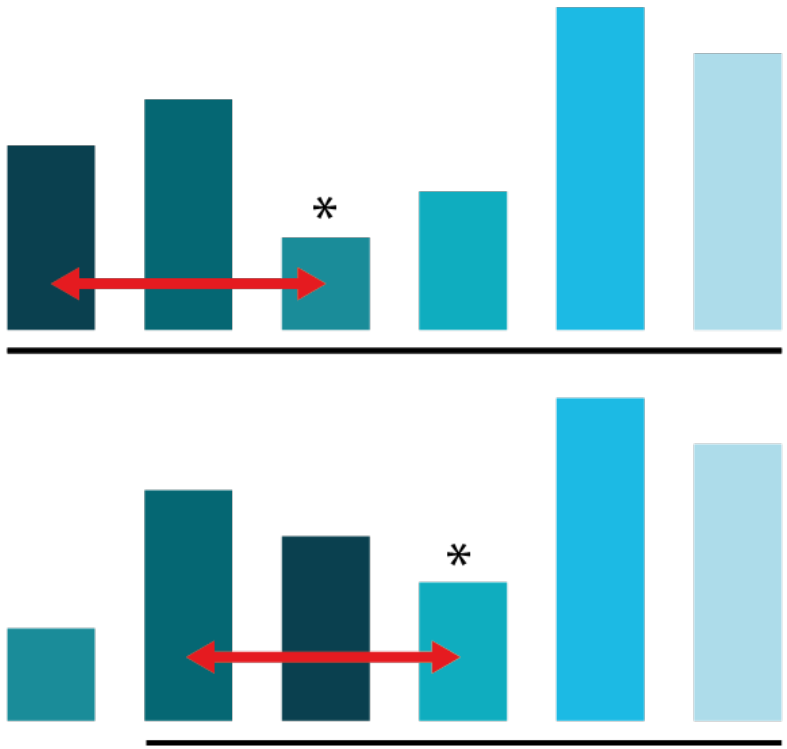


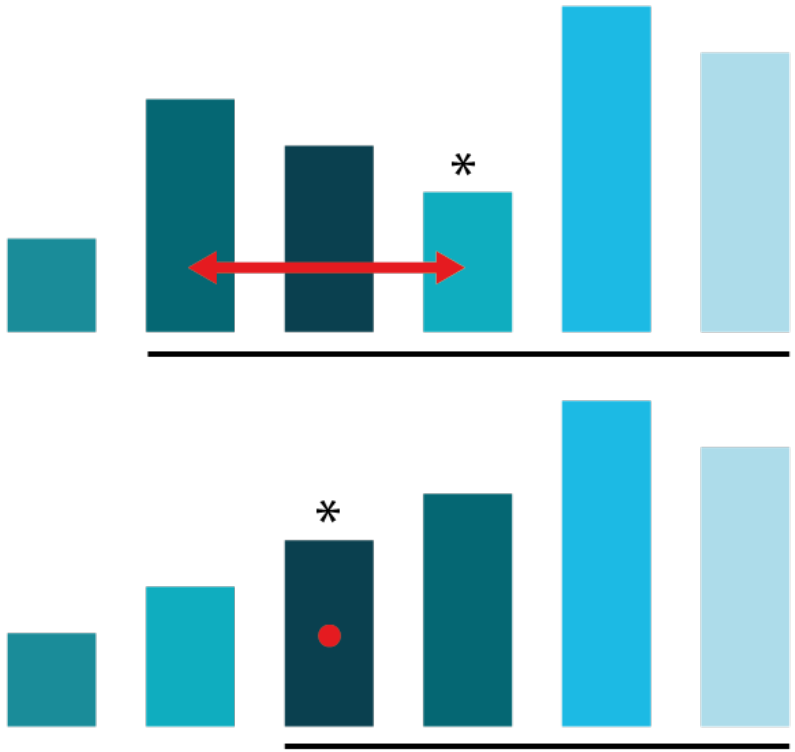
Des objets sont triés selon une relation d'ordre, en lien avec une propriété. Sur la ligne du haut, les rectangles sont organisés selon leur couleur (de la plus sombre à la plus claire), alors que sur la ligne du bas, ils sont triés selon leur taille (du plus petit au plus grand).

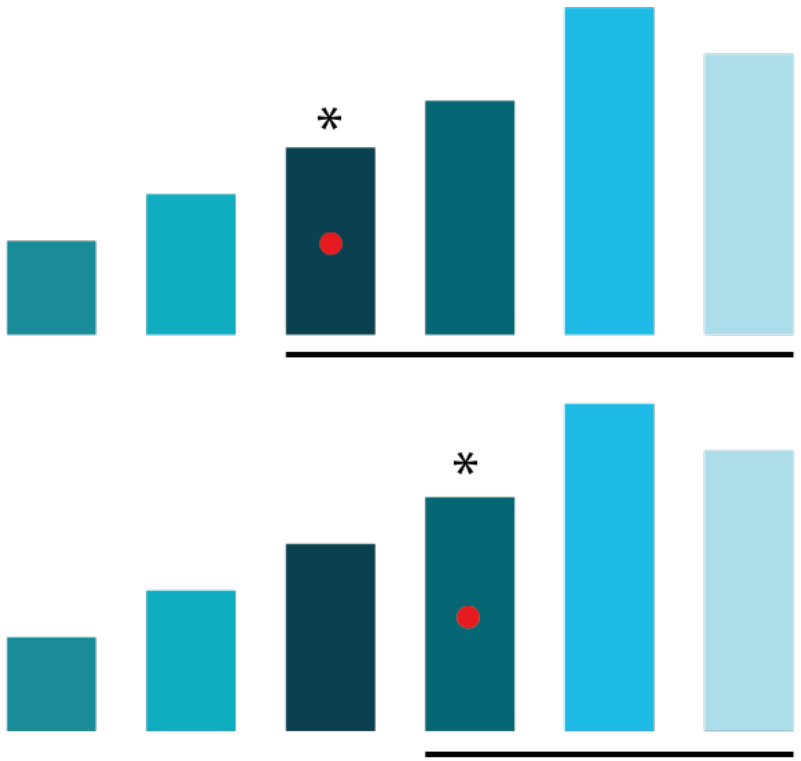
Gymnase Yverdon

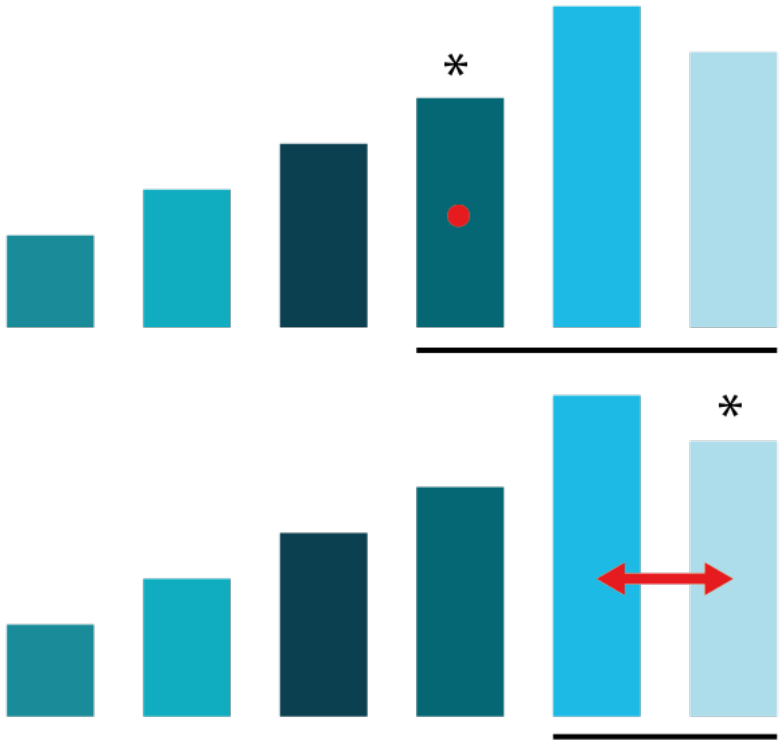
Principe

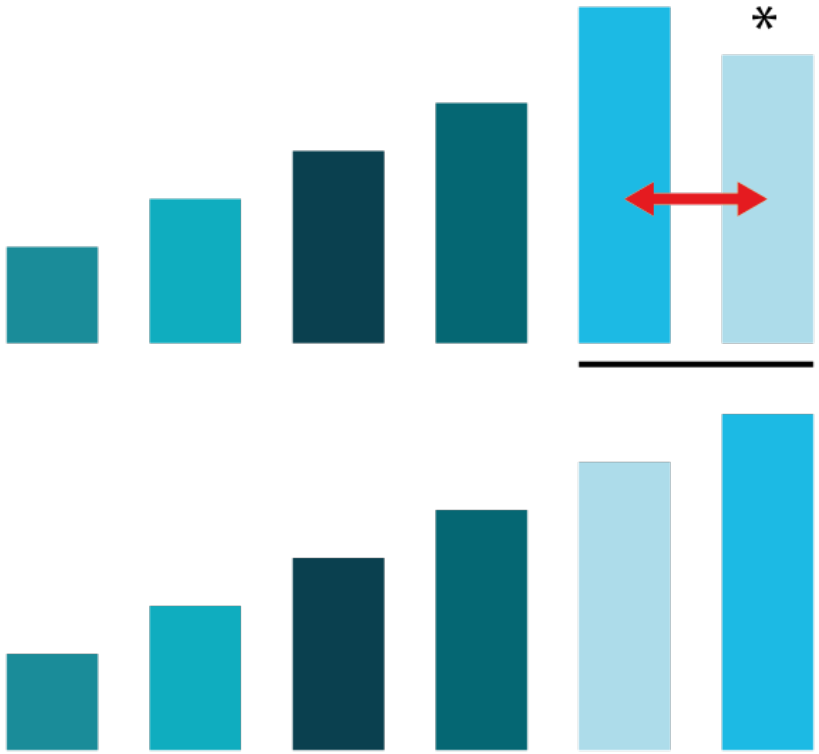
Pour trier dans l'ordre croissant, on **sélectionne** le plus petit élément que l'on échange avec le premier élément de la liste, puis on répète le procédé avec la sous-liste constituée des éléments restants à trier jusqu'à obtenir une sous-liste réduite à un seul élément.











Algorithme 1 : tri par sélection

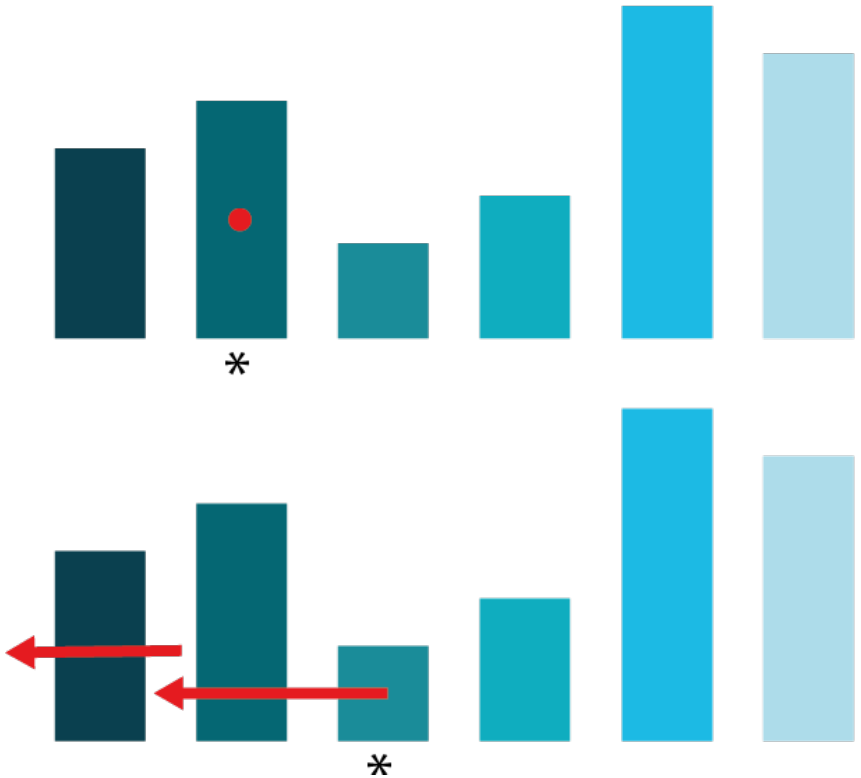
Entrées : un tableau t
Sorties : le tableau trié (dans l'ordre croissant)

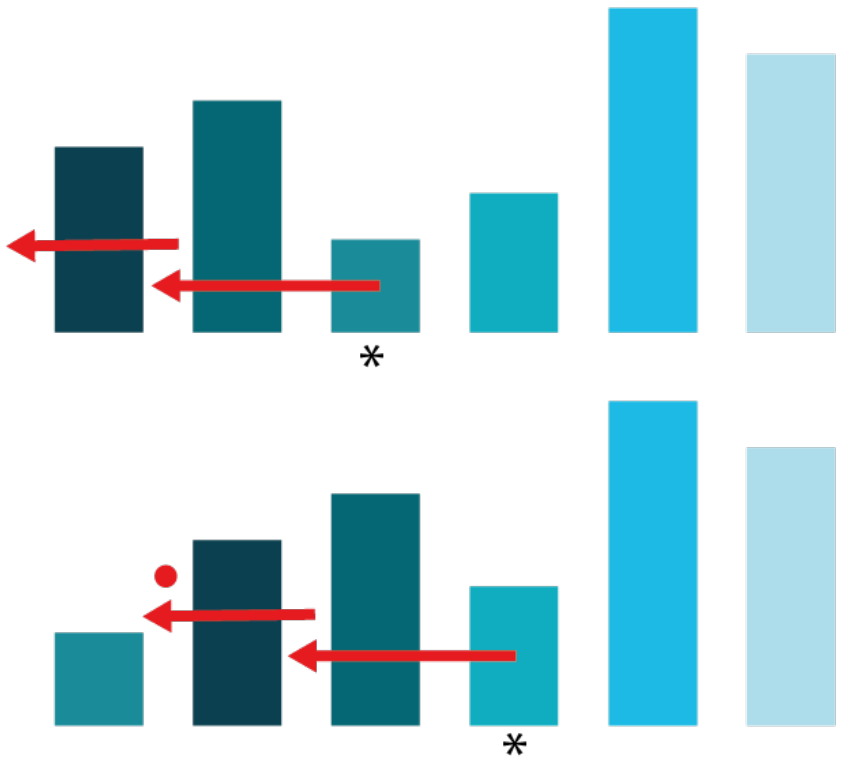
```
n ← longueur(t) ;
pour i de 0 à n-2 (inclus) faire
    // on sélectionne le plus petit élément de la sous-liste non triée
    m ← i ;
    pour j de i+1 à n-1 (inclus) faire
        si t[ j ] < t[ m ] alors
            m ← j
        fin
    fin
    // on échange le 1er élément de la sous-liste non triée et ce plus petit élément
    si m ≠ i alors
        tmp ← t[ i ] ;
        t[ i ] ← t[ m ] ;
        t[ m ] ← tmp ;
    fin
fin
```

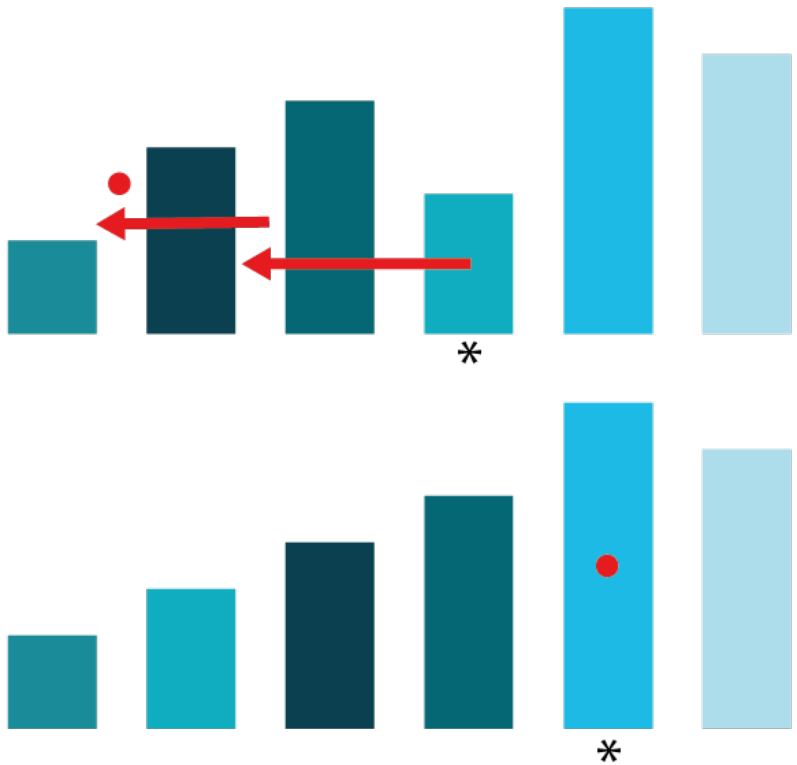
Détailler les différentes étapes du tri par sélection sur la liste [3, 4, 1, 7, 2].

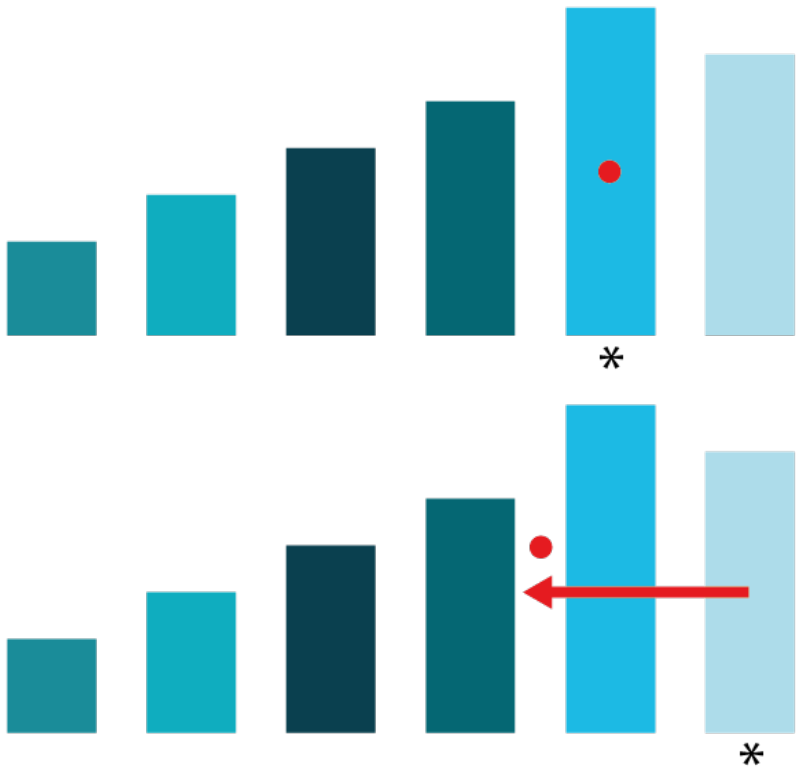
Principe

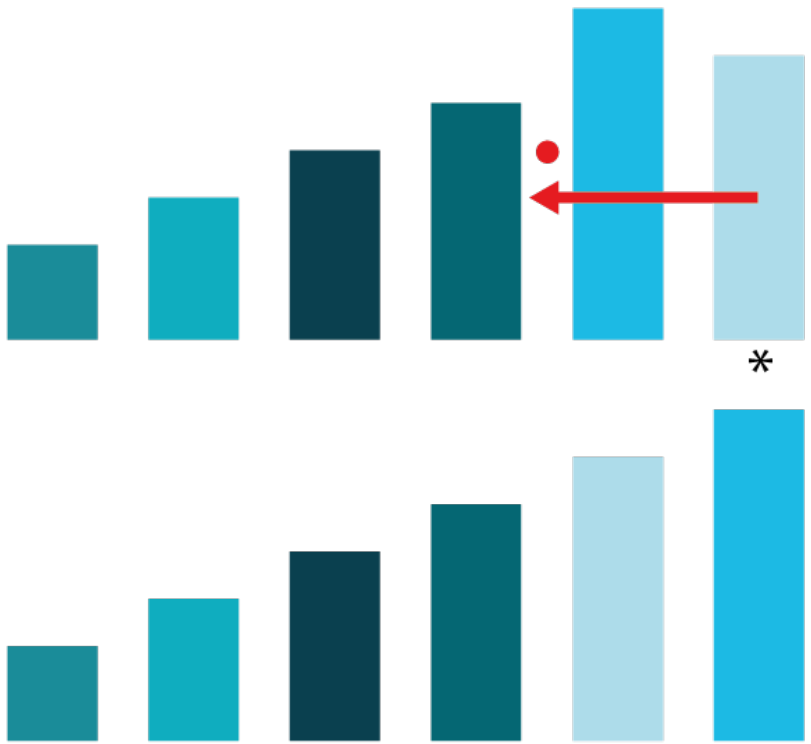
Pour trier dans l'ordre croissant, on **insère** le deuxième élément à la bonne place en le comparant au premier, puis on « insère » le troisième à la bonne place en le comparant aux deux premiers, ...











Algorithme 2 : tri par insertion

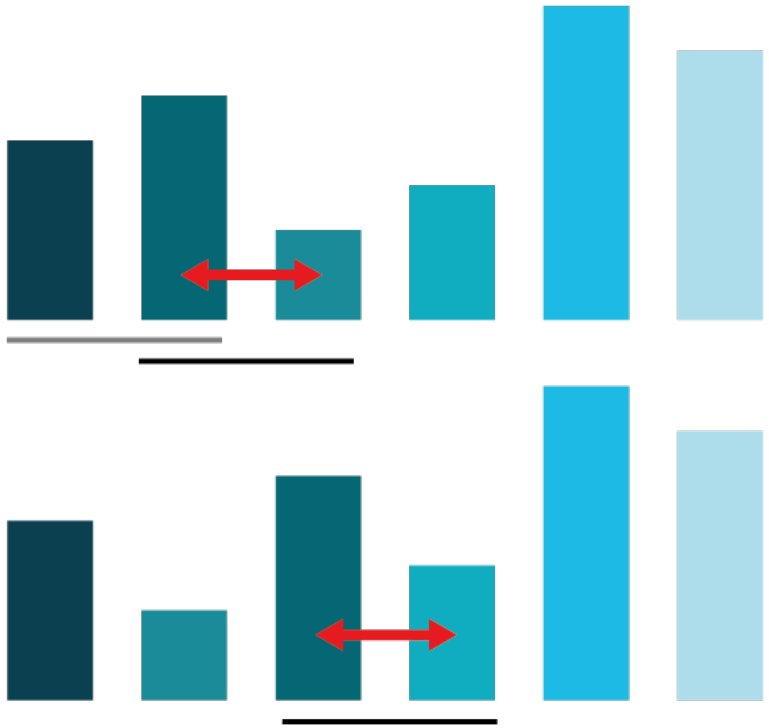
Entrées : un tableau t
Sorties : le tableau trié (dans l'ordre croissant)

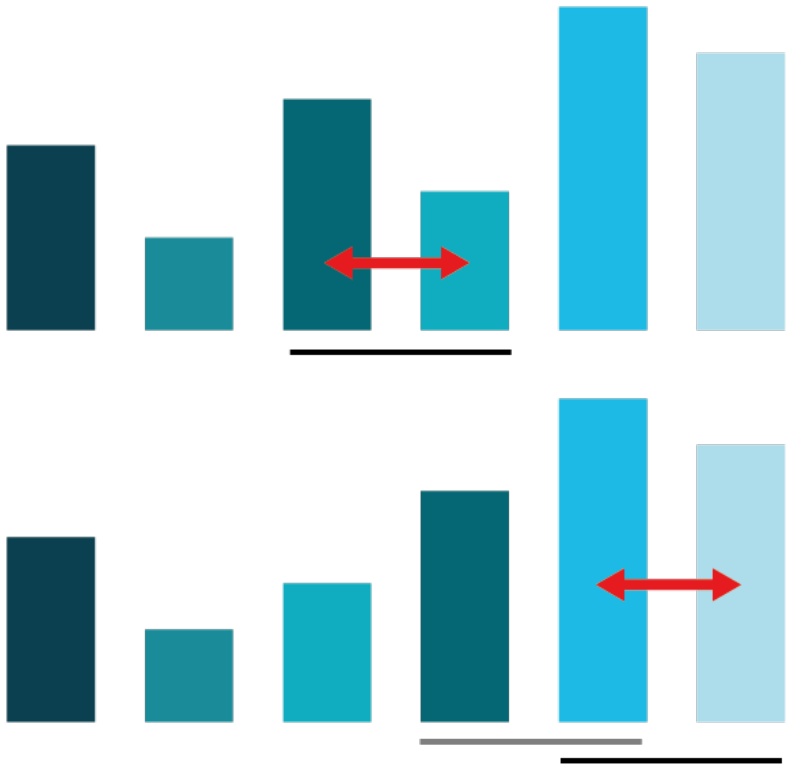
```
n ← longueur(t) ;
pour i de 1 à n-1 (inclus) faire
    // on insère à la bonne place l'élément t[ i ] dans t[ 0..i [ qui est trié
    v ← t[ i ] ;
    j ← i ;
    tant que j > 0 et t[ j-1 ] > v faire
        t[ j ] ← t[ j-1 ] ;
        j ← j-1
    fin
    t[ j ] ← v
fin
```

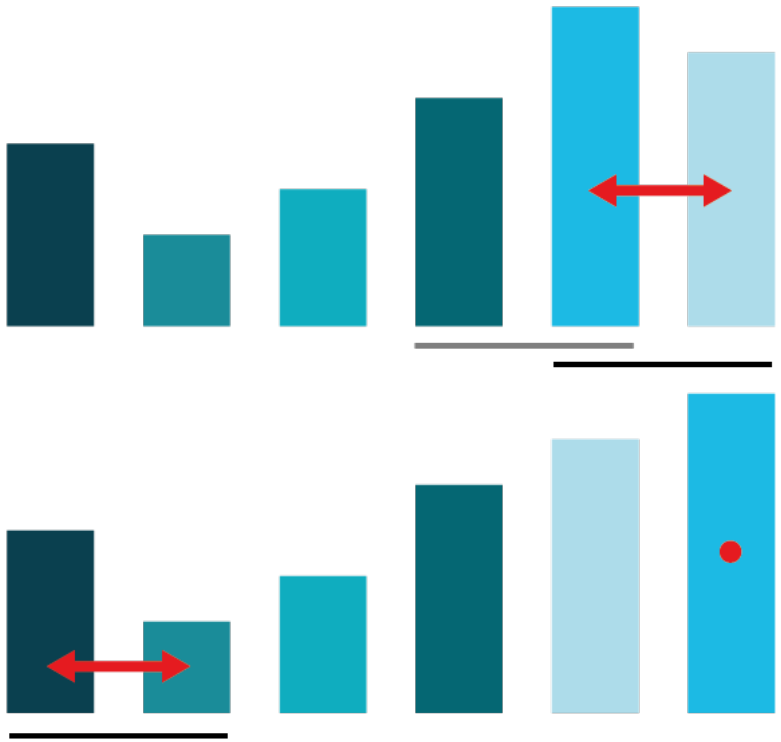
Détailler les différentes étapes du tri par insertion sur la liste [3, 4, 1, 7, 2].

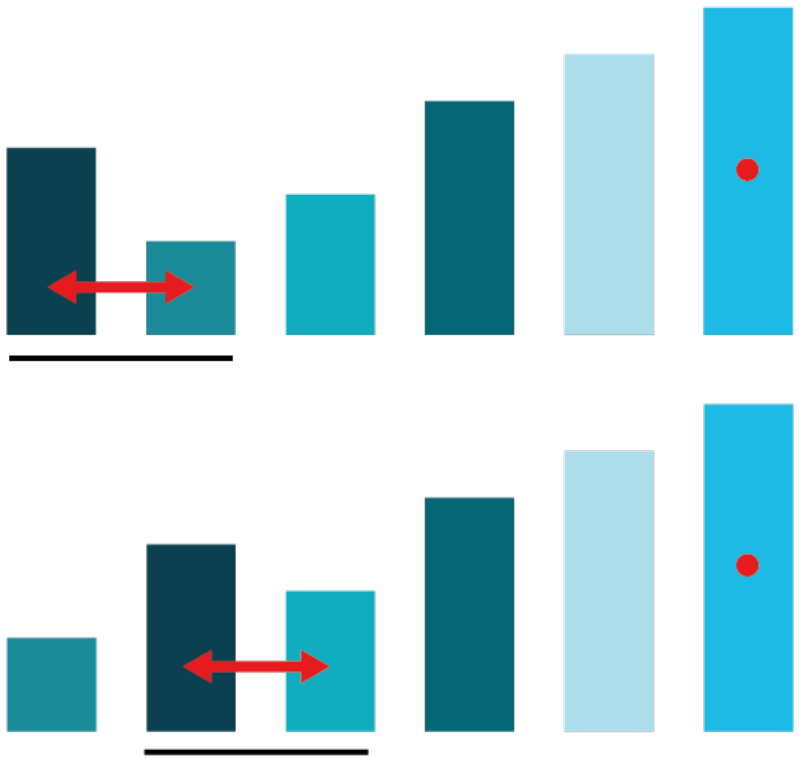
Principe

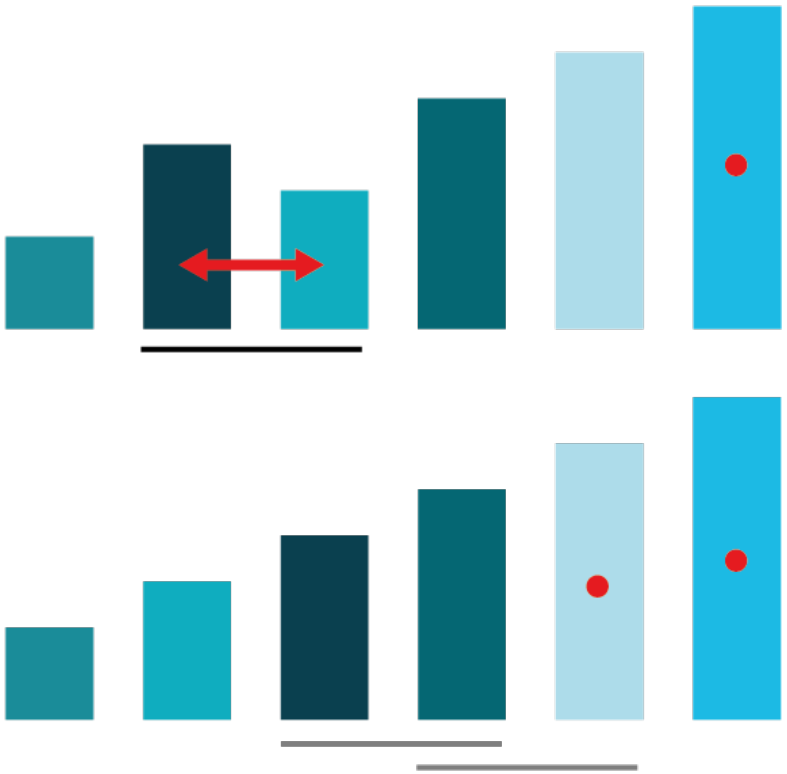
Pour trier dans l'ordre croissant, on trie les éléments voisins, deux à deux, en commençant par les deux premiers jusqu'à la fin de la liste. Après ce premier parcours de la liste, le plus grand élément se retrouve en dernière position, il est « remonté comme une bulle ». Puis on répète le procédé avec la sous-liste constituée des éléments restants à trier jusqu'à obtenir une sous-liste réduite à un seul élément.











Algorithme 3 : tri à bulles

Entrées : un tableau t
Sorties : le tableau trié (dans l'ordre croissant)

```
n ← longueur(t) ;
pour i de 2 à n (inclus) faire
    pour j de 0 à n-i (inclus) faire
        si t[ j ] > t[ j+1 ] alors
            // on échange les deux éléments
            tmp ← t[ j ] ;
            t[ j ] ← t[ j+1 ] ;
            t[ j+1 ] ← tmp ;
        fin
    fin
fin
```

Détailler les différentes étapes du tri à bulles sur la liste [3, 4, 1, 7, 2].

Cette section fait l'objet d'un jupyter notebook disponible sur moodle.

Ce cours s'appuie sur :

- le site internet Modulo développé par les concepteurs du programme
- l'ouvrage Numérique et sciences informatiques aux éditions Ellipses