CERGY PARIS UNIVERSITÉ

浙江科技學院 ZHEJIANG UNIVERSITY OF SCIENCE & TECHNOLOGY

Programming et Algorithme III
Cours II: Tri

Plan du cours

- Propriété de tri (Stabilité, interne/externe)
- Tri
  - Tri par sélection
  - Tri fusion
  - Tri rapide
- Conclusion

Problème du tri

Problème : étant donné un tableau d'entiers *T* , trier *T* dans l'ordre croissant.

- Problème connu
- Grande richesse conceptuelle :
  - Des algorithmes basés sur des idées et des structures de données très différentes. . .
  - Des complexités différentes.
  - Des algorithmes optimaux.

Liste
L = (7, 3, 1, 4, 8, 3)
↓ , tri
L = (1, 3, 3, 4, 7, 8)
Liste triée (en ordre croissant)

Problème du tri

- Tri interne

1)Tri interne: Quand toutes les données sont placées dans la mémoire centrale ou la mémoire interne.
2) Exemples (prèsque tout les exemples dans ce cours):
tri en tas, tri à bulles, tri par sélection, tri rapide, tri en coquille, tri par insertion.
3)utilisé pour les petites quantités de données

  - éléments en table, liste chaînée. En mémoire centrale

- Tri externe

1) Tri externe: toutes les données à trier ne peuvent pas être placées en même temps dans la mémoire.
2) Exemples: Le tri par fusion et ses variantes
3) utilisé pour les quantités massives de données
4) Certains supports de stockage externes, tels que les disques durs et les CD, sont utilisés pour le tri externe

  - éléments en fichiers

- Opérations élémentaires

  - comparaison de deux éléments
  - échange
  - sélection de places

Stabilité du tri

L = (e1, e2, …, en) en table, accès direct à e_i

Clé : Élément → Ensemble muni de l 'ordre ≤

Problème

Calculer *p*, permutation de *{1, 2, …, n}*

telle que *Clé*($e_{p(1)}$) ≤ Clé($e_{p(2)}$) ≤ … ≤ Clé($e_{p(n)}$)

Rang

$p^{-1}(i)$ est le rang de l'élément *i* dans la suite classée

Stabilité

p est stable, chaque fois que *Clé* ($e_{p(i)}$) = *Clé* ($e_{p(k)}$) :

$i < k$ équivalent $p(i) < p(k)$

[Traduction: le tri n'échange pas les éléments de même clé]

Stabilité du tri

Il y des question à étudier :

- La définition de stabilité (c’est quoi un algorithme stable ?)
- Pourquoi la stabilité est-elle une propriété utile des algorithmes de tri ? (la stabilité sert à quoi?)
- Comment on peut vérifier si un algo est estable ?

c’est quoi un algorithme stable ?

1) Regarder cette vidéo
2) an exemple

Quel algorithme est stable ?

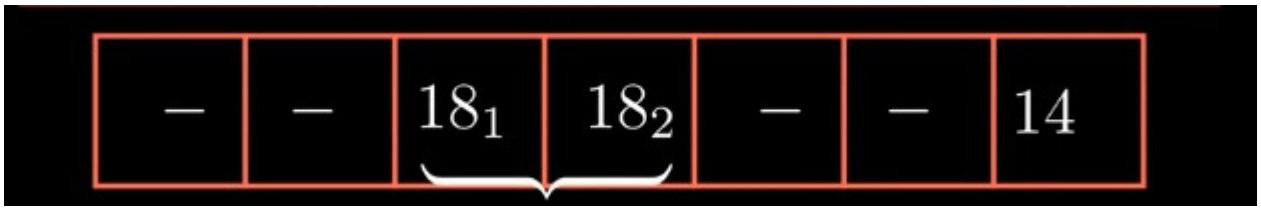
1) Any sorting algorithm has stable version but there can be additional time/space complexities

2) les algorithmes avec la condition suivante ne sont pas du tout stable :

les **permutations d'éléments non adjacents** sont la cause première de l'instabilité des algorithmes de tri

Quel algorithme est stable ?

1) Si l'algorithme peut seulement permuter les éléments adjacents il est stable



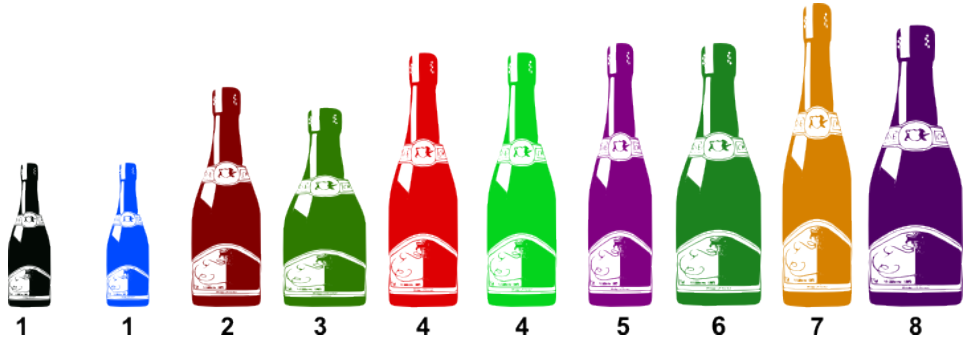
2) Exemples :

- **Le tri par merge, par insertion**
  pas de permutation(swaps) → stable
- **Le tri à bulles :**
  Permutation des eléments adjacents → stable
- **Le tri rapide, et par selection :**
  Permutation des eléments non-adjacents → non-stable

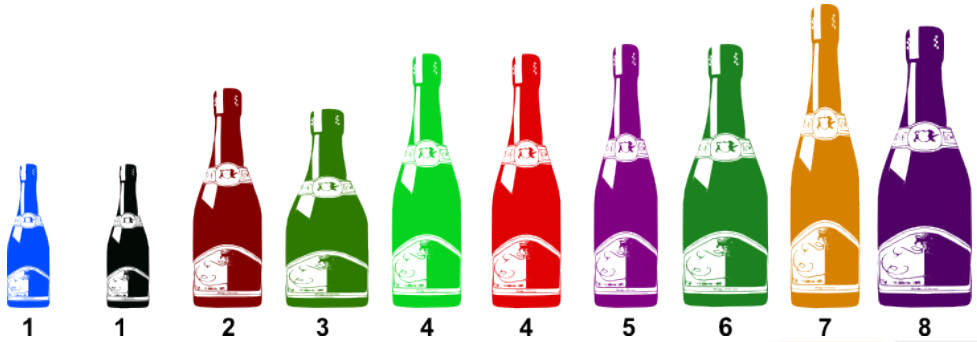
Stabilité du tri: exemple plus vive

Trier la collection de bouteilles ci-dessous par ordre de volume (indiqué sous la bouteille):

Si vous obtenez ceci, alors votre tri n'était pas stable :
(la bouteille noire de volume 1 se trouve maintenant avant la bouteille bleue de même volume alors qu'elle devrait être après)



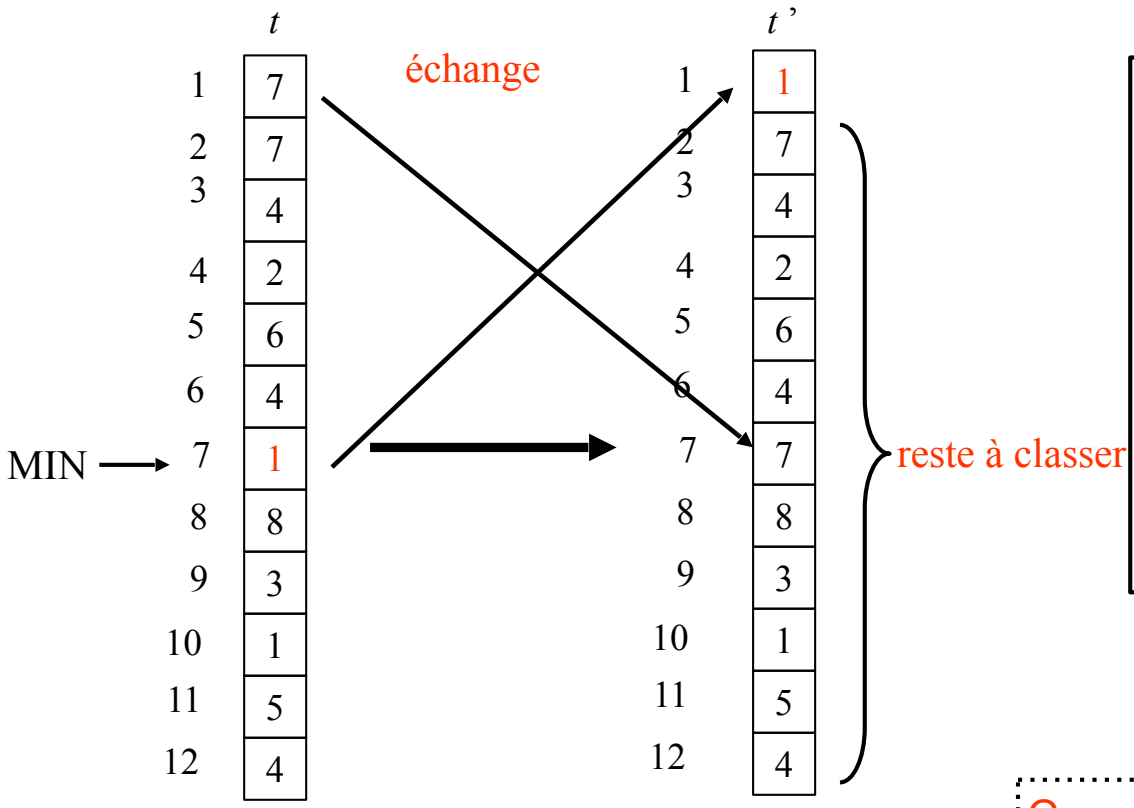
Avec un tri stable, on aurait obtenu :



CERGY PARIS UNIVERSITÉ

浙江科技學院 ZHEJIANG UNIVERSITY OF SCIENCE & TECHNOLOGY

Tri par sélection




```
fonction  TRI_PAR_SELECTION (t table [1…n]) : table ;
début
        pour  i ← 1  à  n-1  faire
        {  min ← i ;
           pour  j ←  i + 1  à  n  faire
                 si  t [ j ]  < t [ min ]  alors  min ← j ;
           temp ← t  [ i ] ;
           t [ i ] ← t [ min ] ;
           t [ min ] ← temp ;
        }  retour  ( t ) ;
fin .
```

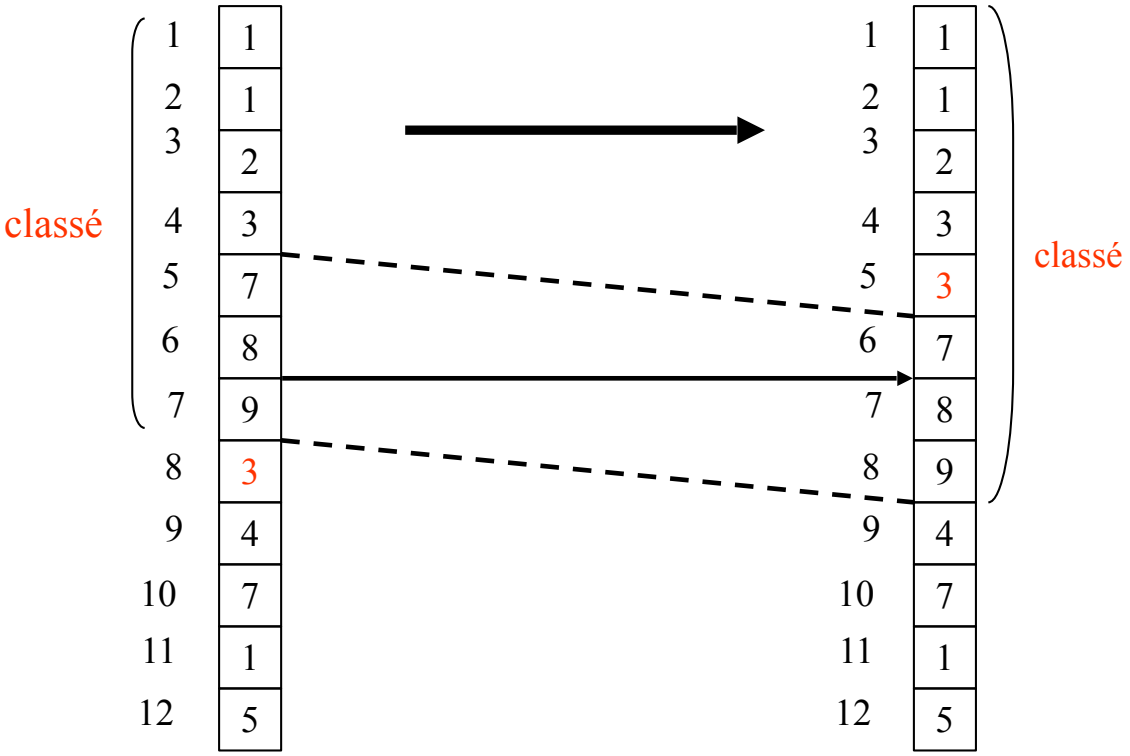
Recherche du minimum par balayage séquentiel

Ou organisation des éléments en « tas » (voir « file de priorité »)

Complexité : espace O (1)
temps O (n^2) { O (n^2) comparaisons ; n-1 échanges

Stabilité: Dépend d'implémentation

Tri par insertion




```
fonction TRI_ PAR_INSERTION ( t table
[1...n]) : table ;

début t [ 0 ] ← – ∞ ;
        pour i ← 2 à n faire
        { k ← i - 1 ; temp ← t [ i ] ;
          tant que temp < t [ k ] faire
          { t [ k + 1] ← t [ k ] ; k ← k - 1 ; }
            t [ k + 1] ← temp ;
        }
        retour ( t ) ;
fin .
```

Point d ’insertion
- recherche séquentielle
- recherche dichotomique

Complexité : espace O (1)
temps O (n^2) { O (n^2) comparaisons ; O (n^2) affectations d’éléments

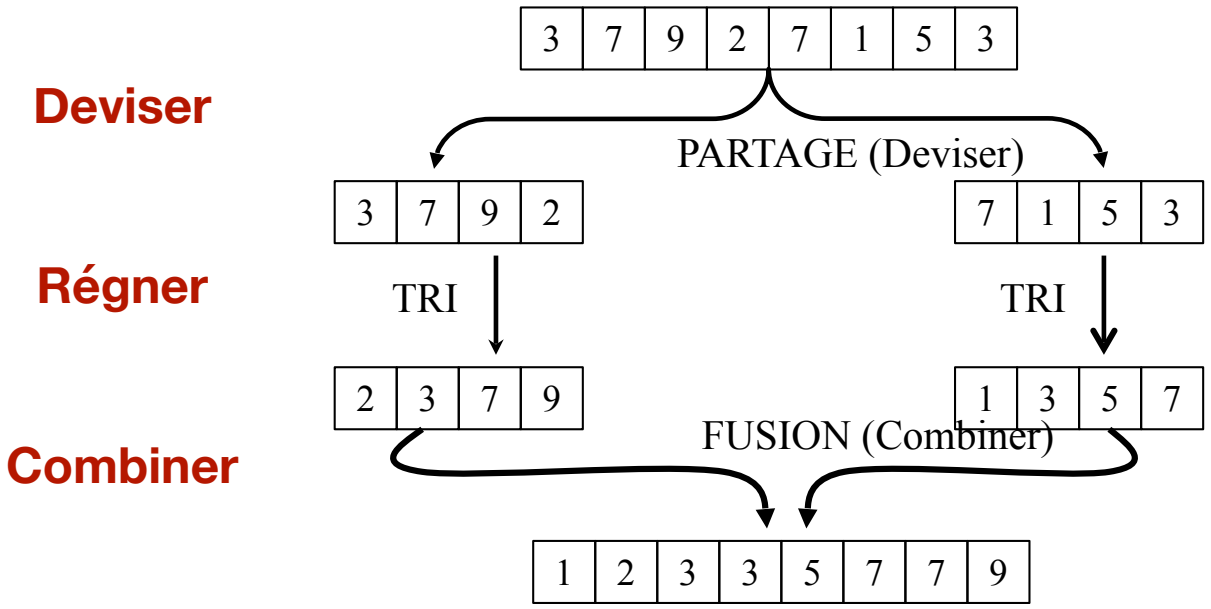
Insertion dichotomique : O (n log n) comparaisons
Stabilité: stable

CERGY PARIS UNIVERSITÉ

浙江科技學院 ZHEJIANG UNIVERSITY OF SCIENCE & TECHNOLOGY

Tri fusion

Une stratégie de diviser pour régner



Pseudo-code de tri fusion

```
fonction  TRI_Fusion ( t  table [1…n] ) : table ;
    si n ≤ 1
        renvoyer t
    sinon
        renvoyer fusion(TRI_Fusion(T[1, …, n/2]), TRI_Fusion(T[n/2 + 1, …, n]))
```

```
entrée : deux tableaux triés A et B
sortie : un tableau trié qui contient exactement les éléments des tableaux A et B
fonction fusion(A[1, …, a], B[1, …, b])
    si A est le tableau vide
        renvoyer B
    si B est le tableau vide
        renvoyer A
    si A[1] ≤ B[1]
        renvoyer A[1] ⊕ fusion(A[2, …, a], B)
    sinon
        renvoyer B[1] ⊕ fusion(A, B[2, …, b])
```

Stable: Les elements équivalents sont copiés sans alterner