Terminale NSI - Algorithmique

Diviser pour régner. Tri fusion, présentation

2020/04/26

Tri fusion

On applique diviser pour régner pour trier un tableau.

Même principe :

Tri Fusion (`tableau`):

- Si `tableau` est de taille $<= 1$ on ne fait rien.
- Sinon, On sépare `tableau` en 2 parties `gauche` et `droite`,
- On appelle Tri fusion sur `gauche` et sur `droite`
- On fusionne `gauche` et `droite` dans `tableau`

```
fusionner (`tableau`, `gauche`, `droite`):

* On parcourt les deux tableaux `gauche` et `droite` en mêm
    Pour chaque paire d'éléments, on place le plus petit da
* S'il reste des éléments dans `gauche` ou dans `droite` or
    de tableau
```

Exemple simple

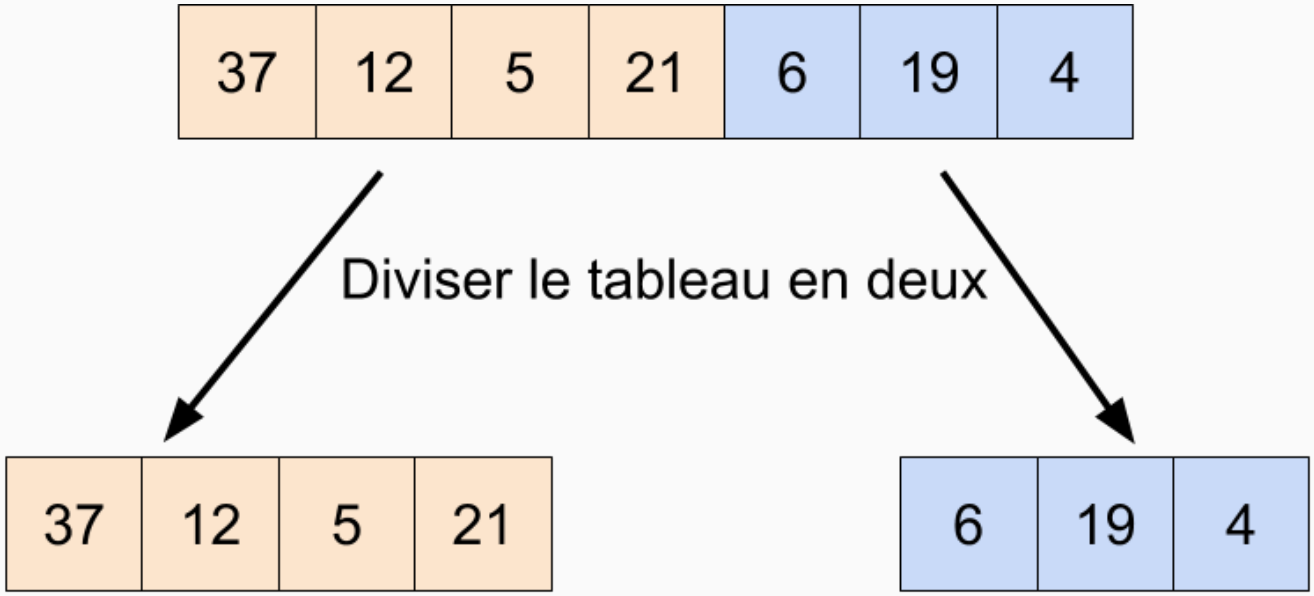
Vidéo Geek for geek

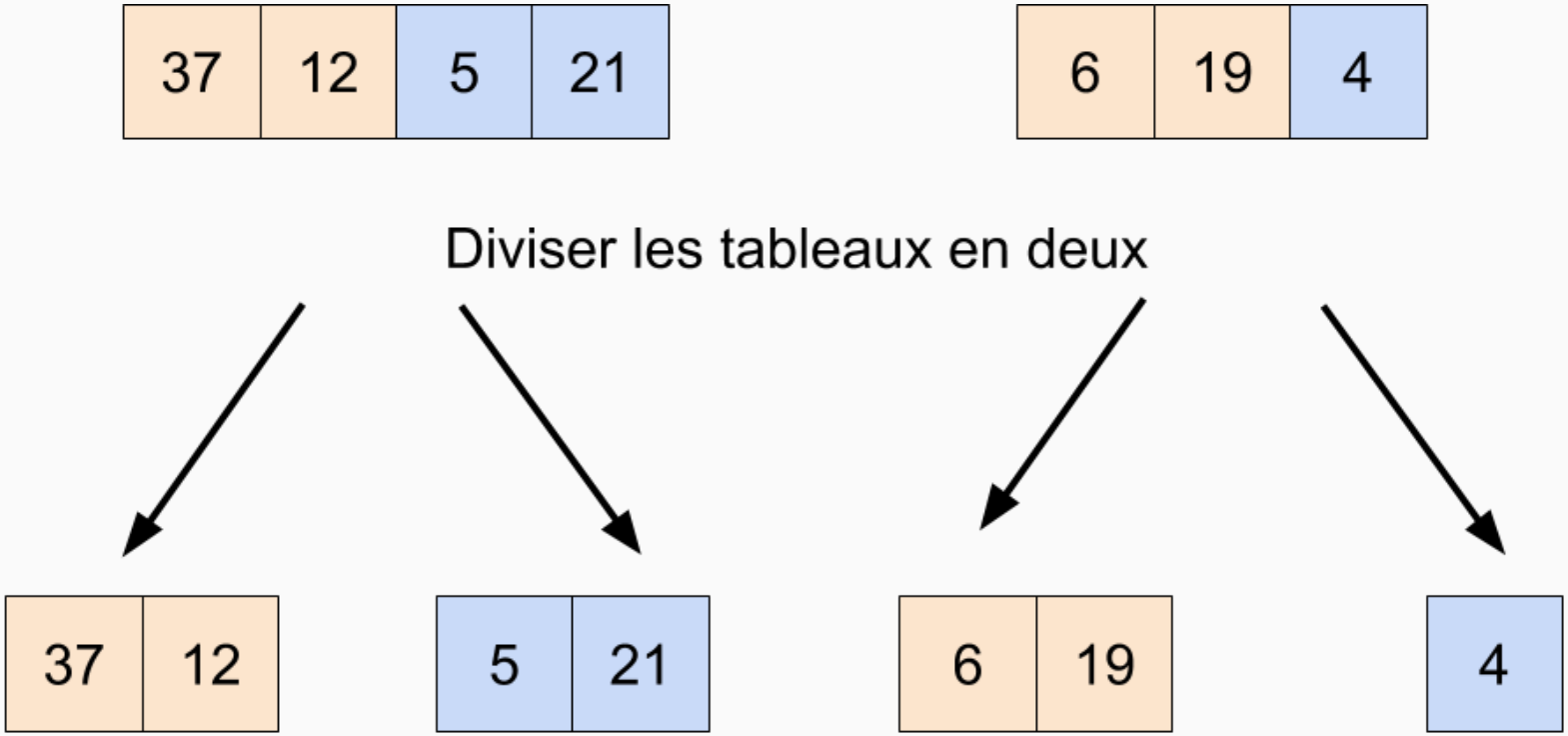
Exemple détaillé 0

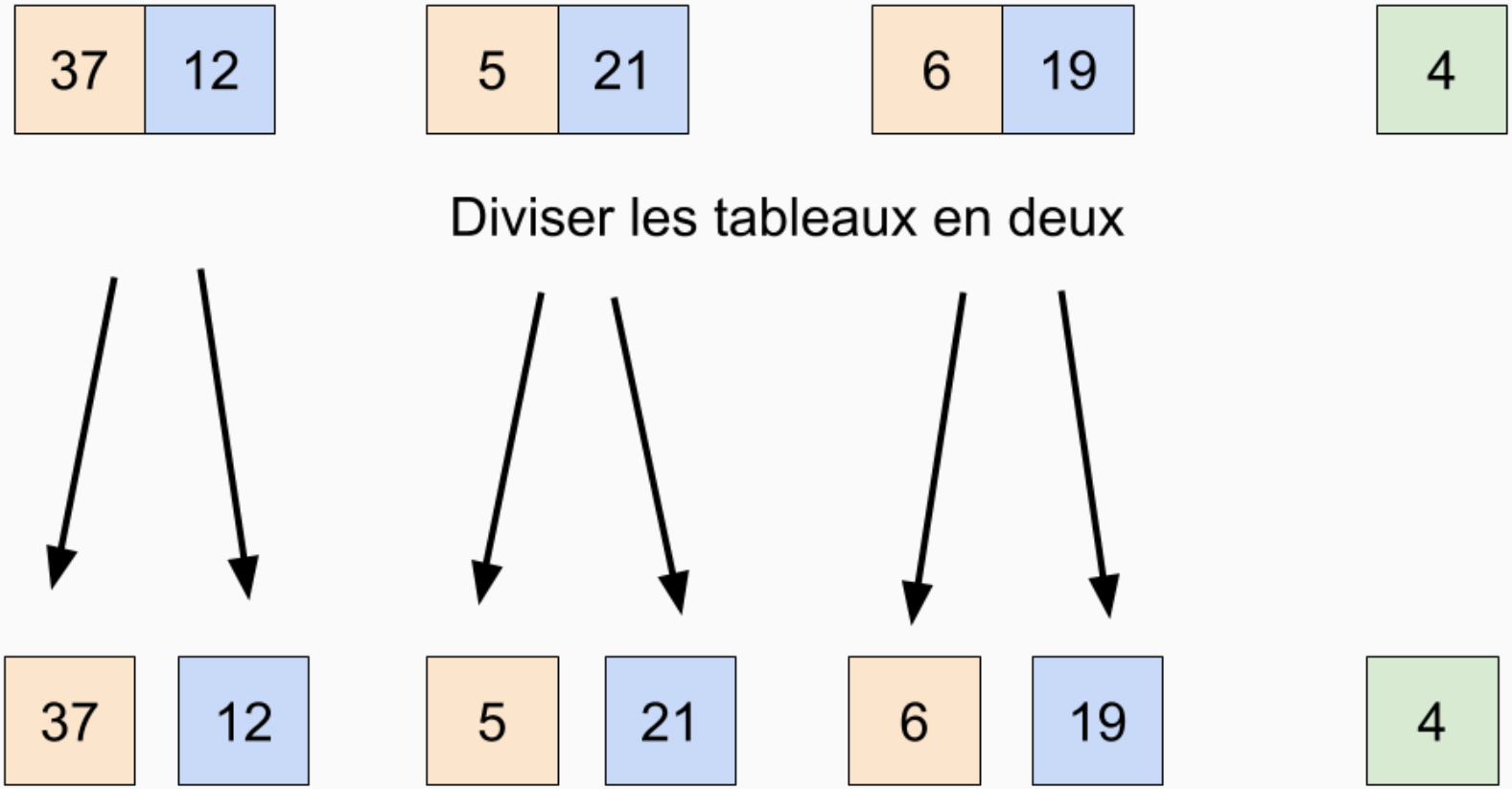
37	12	5	21	6	19	4

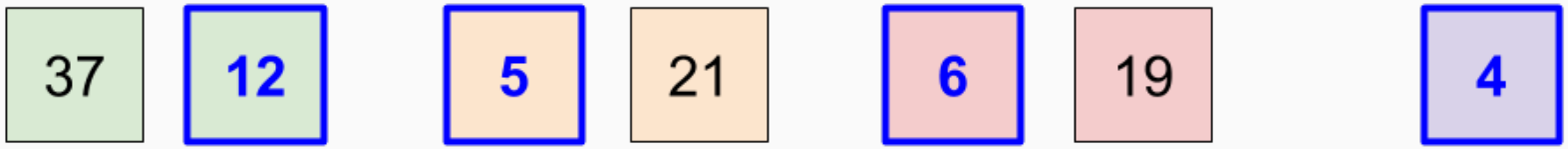
Tri fusion de ce tableau

.

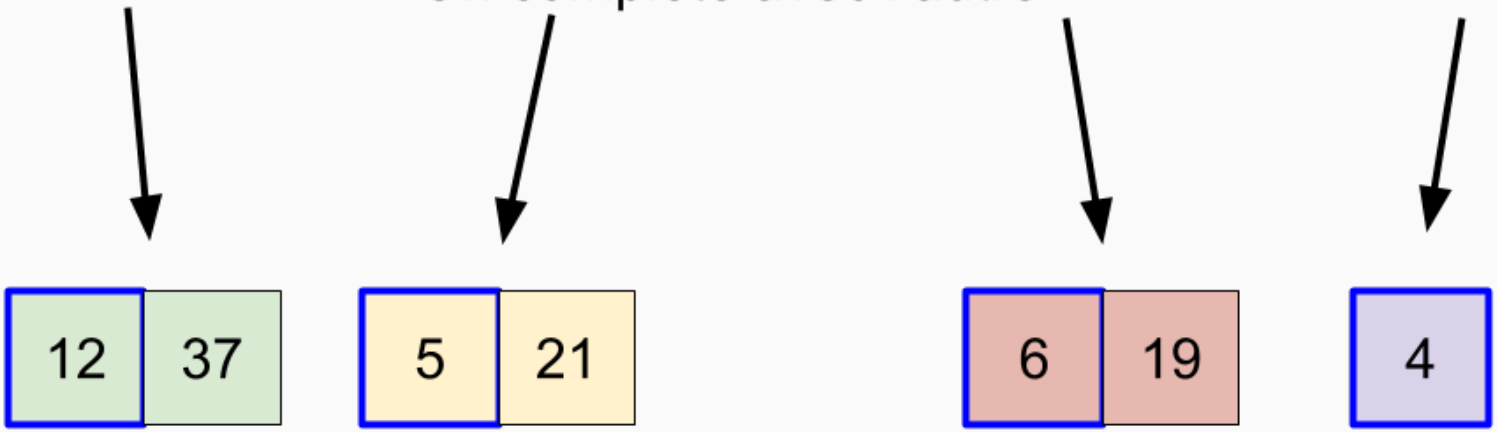

37
12
5
21
6
19
4
Diviser le tableau en deux
37
12
5
21
6
19
4


37
12
5
21
6
19
4
Diviser les tableaux en deux
37
12
5
21
6
19
4


37
12
5
21
6
19
4
Diviser les tableaux en deux
37
12
5
21
6
19
4



Fusionner :
On choisit le plus petit des deux
On complète avec l'autre



Fusionner :

On parcourt les tableaux par paire.
Chaque fois, le plus petit d'une paire d'éléments est choisi.

On complète avec ce qui reste

5	12	21	37

4	6	19

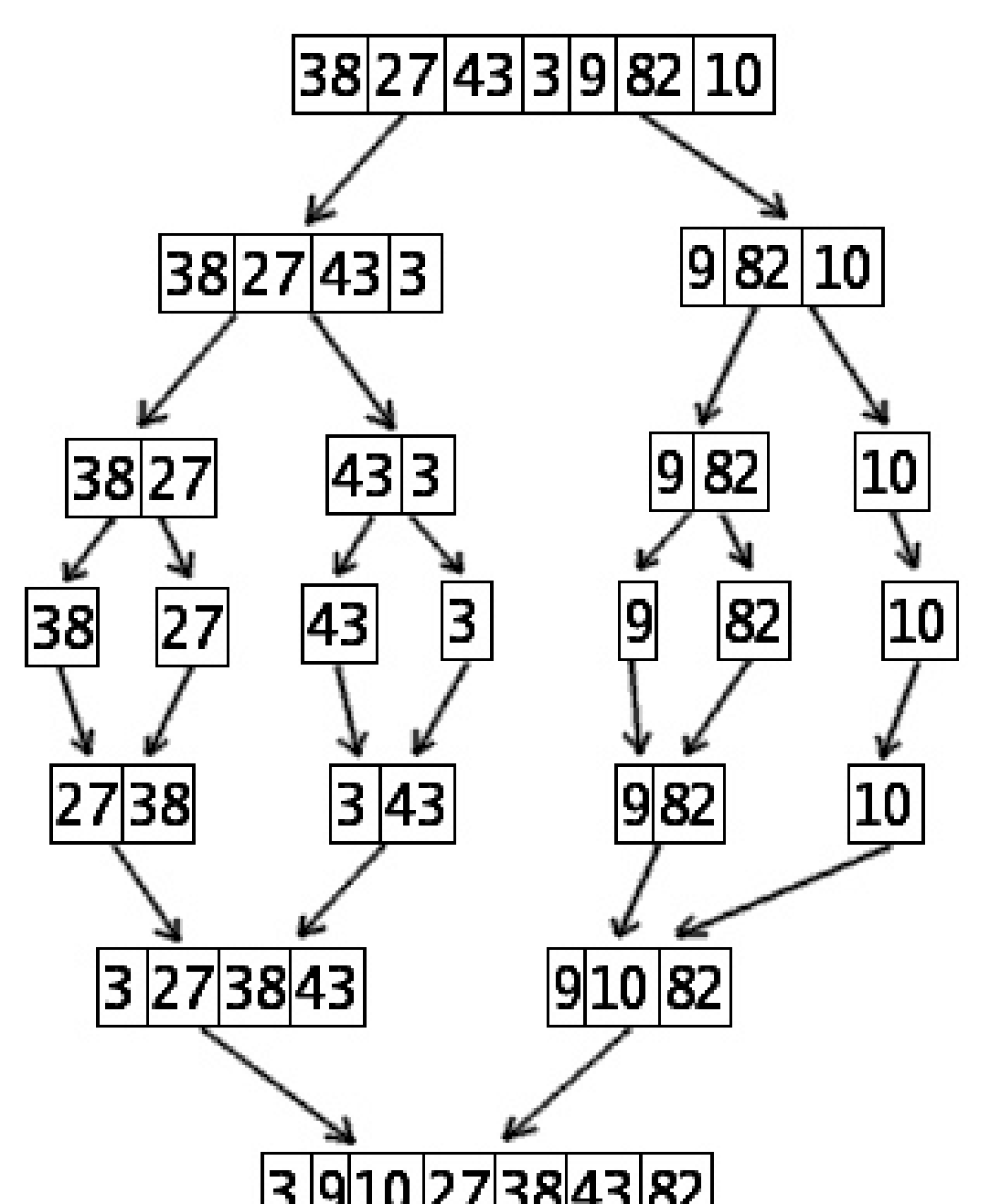
Fusionner :

On parcourt les tableaux par paire.
Chaque fois, le plus petit d'une paire d'éléments est choisi.

On complète avec ce qui reste

4	5	6	12	19	21	37

En une seule image



Algorithme : tri_fusion

```
tri_fusion (tableau) :
    Si la longueur est > 1:
      # séparer
      milieu = longueur // 2
      gauche = tableau [0 ... milieu - 1]
      droite = tableau [milieu ... fin]

      # diviser
      tri_fusion(gauche)
      tri_fusion(droite)

      # combiner
      fusion(tableau, gauche, droite)
```

Algorithme Fusion

```
fusion (tableau, gauche, droite)
  i, j, k = 0
  tant que i < longueur de gauche et j < longueur de droite
      Si gauche[i] < droite[j] alors
          tableau[k] = gauche[i] et i = i + 1
      Sinon
          tableau[k] = droite[j] et j = j + 1
      k = k + 1

  Pour les éléments restant, les ajouter à fin de tableau
```

Exemple détaillé pour la fusion

Fusionner `g = [1, 2, 5]` et `d = [3, 4]` : le premier élément du tableau fusionné sera le premier élément d'une des deux tableaux d'entrée (soit 1, soit 3) car ce sont des listes triées.

`g = [1, 2, 5]` et `d = [3, 4]`

- Comparer 1 et 3 $\rightarrow$ 1 est le plus petit.

  `t=[1]`

`g = [1, 2, 5]` et `d = [3, 4]`

- Comparer 1 et 3 $\rightarrow$ 1 est le plus petit.

  `t=[1]`

- Comparer 2 et 3 $\rightarrow$ 2 est le plus petit.

  `t=[1, 2]`

`g = [1, 2, 5]` et `d = [3, 4]`

- Comparer 1 et 3 $\rightarrow$ 1 est le plus petit.

  `t=[1]`

- Comparer 2 et 3 $\rightarrow$ 2 est le plus petit.

  `t=[1, 2]`

- Comparer 5 et 3 $\rightarrow$ 3 est le plus petit.

  `t=[1, 2, 3]`

`g = [1, 2, 5]` et `d = [3, 4]`

- Comparer 1 et 3 $\rightarrow$ 1 est le plus petit.

  `t=[1]`

- Comparer 2 et 3 $\rightarrow$ 2 est le plus petit.

  `t=[1, 2]`

- Comparer 5 et 3 $\rightarrow$ 3 est le plus petit.

  `t=[1, 2, 3]`

- Comparer 5 et 4 $\rightarrow$ 4 est le plus petit.

  `t=[1, 2, 3, 4]`

`g = [1, 2, 5]` et `d = [3, 4]`

- Comparer 1 et 3 $\rightarrow$ 1 est le plus petit.

  `t=[1]`

- Comparer 2 et 3 $\rightarrow$ 2 est le plus petit.

  `t=[1, 2]`

- Comparer 5 et 3 $\rightarrow$ 3 est le plus petit.

  `t=[1, 2, 3]`

- Comparer 5 et 4 $\rightarrow$ 4 est le plus petit.

  `t=[1, 2, 3, 4]`

- Compléter 5 $\rightarrow$ `t=[1, 2, 3, 4, 5]`

Récursion

Il faut bien comprendre l'ordre dans lequel sont effectués les appels récursifs

`tri_fusion` s'appelle elle même **AVANT** d'appeler `fusion`

Donc :

D'abord on découpe le tableau, sa première moitié, son premier quart etc. jusqu'à avoir un élément (le premier).

Ensuite on fait fusion sur lui (il ne change pas) Ensuite on arrive à la fusion des deux premiers,

Pour avancer il faut avoir découpé (3 et 4) jusqu'à avoir une taille 1, Ensuite on les fusionne.

Et on fusionne les éléments 0, 1, avec 2, 3.

etc.

L'algorithme ne progresse pas "par étage" comme la présentation le laisse croire.

Complexité

- La partie “diviser” est de complexité constante.
- Combien d’étapes dans le tri fusion ? Autant d’étape qu’il en faut pour arriver à `log_2(n)` en effectuant des divisions par 2.

Exemple

Pour un tableau de taille `n = 64` il faut :

$64/2 = 32, 32/2 = 16, 16/2 = 8, 8/2 = 4, 4/2 = 2, 2/2 = 1$:

6 étapes.

$2^6 = 64$.

Comme toujours quand on peut séparer le tableau en deux, la méthode diviser pour régner permet de ne réaliser que $\log_2 n$ étapes. Mais. . .

Complexité fusion

- La partie fusion utilise une boucle qui parcourt plusieurs tableaux en même temps.
  - On réalise à chaque étape la même chose :
    - lire deux valeurs,
    - comparer,
    - ranger la plus petite.

  La complexité est linéaire.

Utilisation du tri fusion

Contrairement au tri par sélection ou par insertion, le tri fusion est réllement utilisé en pratique : C++ et java

Il a de nombreux avantages :

- complexité optimale (cela ne signifie pas qu'il est le plus rapide)
- stable (voir plus bas)
- facile à mettre en oeuvre

Cependant, il est possible d'améliorer la méthode :

`timsort`, le tri natif en Python et Javascript utilise une combinaison du tri fusion et du tri par insertion.