S1.02

SAE S1.02

Etude de quelques algorithme de tri incontournables

dernière mise à jour le 16/12/2021
SAE S1.02 / B. Desbenoit

ETUDE DE QUELQUES ALGORITHMES DE TRI INCONTOURNABLES

L'objectif de ce cours est d'étudier quelques algorithmes de tris souvent utilisés en informatique. On s'attachera en particulier à étudier leurs comportements et à comparer leurs performances en fonction de la taille de l'ensemble à trier.

TRI SÉLECTION

SAE S1.02

TRI SÉLECTION

PRINCIPE

Sur un tableau de n éléments, le tri par sélection consiste à : rechercher le plus petit élément du tableau, et l'échanger avec l'élément d'indice 1 ; puis, rechercher le second plus petit élément du tableau, et l'échanger avec l'élément d'indice 2 ; continuer de cette façon jusqu'à ce que le tableau soit entièrement trié.

Cela revient en fait à déterminer la place d'un élément en comptant combien il y a dans le tableau d'éléments plus petits, aux doublons prêt.

ALGORITHME

1. On se place à la première case du tableau.
2. Pour l'élément courant, on cherche parmi les éléments à sa droite, un plus petit que lui et si on en trouve un, on les permute.
3. On avance d'une case tant que l'on n'a pas atteint la fin du tableau.

```
Procedure TriSelection(↔tab : tableau [TMAX]
Entier
                                , →nbElements :
Entier)
Variable
  m : entier
  i, j: entier
Début
```

```
  pour i de 0 à nbElements-1 faire
    m ← i
    pour j de i+1 à nbElements-1 faire
      Si (tab[j] < tab[m] ) alors
        m ← j
      Finsi
    FinPour
    permute(tab[i], tab[m])
  FinPour
Fin
```

TRI SÉLECTION

TRI À BULLES

SAE S1.02

TRI À BULLES

PRINCIPE

Le tri à bulles est un algorithme de tri qui repose sur la comparaison des couples d'éléments successifs. A chaque fois que deux éléments qui se suivent ne sont pas rangés dans l'ordre croissant, on les permute. Ainsi, les valeurs donnent l'impression de remonter dans le tableau, un peu comme des bulles dans une boisson gazeuse. Ce tri est aussi appelé tri par propagation.

ALGORITHME

1. L'algorithme parcourt le tableau de la case 0 à la case $n - i$ où i est le nombre d'éléments déjà triés, et compare les couples d'éléments successifs.
2. Lorsque deux éléments successifs ne sont pas dans l'ordre croissant, ils sont échangés.
3. Après chaque parcours du tableau, l'algorithme recommence l'opération.
4. Lorsqu'aucun échange n'a lieu pendant un parcours, cela signifie que le tableau est trié.
5. On arrête alors l'algorithme.

TRI À BULLES

ALGORITHME

```
Procedure TriBulle(↔tab tableau[TMAX]
Entier, →nbElements : Entier)
Variable
  i, j : entier
Début
  pour i de nbElements-1 à 1 pas de -1 faire
    pour j de 1 à i faire
      Si (tab[j-1] > tab[j])
        swap(tab[j-1], tab[j])
      Finsi
    FinPour
  FinPour
Fin
```

```
Procédure TriBulle(↔tab tableau[MAX] Entier,
→nbElements : Entier)
Variable
  i, j : Entier
  bool : Booléen
Début
  i ← nbElements
  bool ← vrai
  TantQue (i≥1 ET bool)
    Pour j de 1 à i faire
      bool ← (tab[j-1] > tab[j])
      Si (bool) Alors
        swap(tab[j-1], tab[j])
      FinSi
    FinPour
    i ← i – 1
  FinTantQue
Fin
```

TRI À BULLES

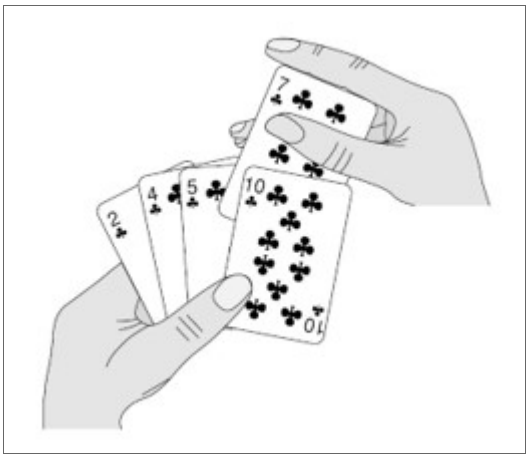
TRI INSERTION

SAE S1.02

TRI PAR INSERTION

PRINCIPE

Le tri par insertion correspond à la méthode utilisée pour trier un paquet de cartes. On prend une carte, puis 2 et on les met dans l'ordre si nécessaire, puis 3 et on met la 3ème carte à sa place dans les 2 premières, etc. De manière générale on suppose les $(i-1)$ premières cartes triées. On prend la i^e carte, et on essaie de la mettre à sa place dans les $(i-1)$ cartes déjà triées. Et on continue jusqu'à $i = n$.



TRI PAR INSERTION

ALGORITHME

1. On parcourt le tableau à trier du deuxième élément à la fin.
2. On compare chaque élément e_i, situé à la i^e place à son prédécesseur.
3. Si e_i est plus petit, on le fait glisser vers la gauche tant qu'on ne trouve pas un élément plus petit ou que l'on n'a pas atteint le début du tableau.
4. Sinon on passe à l'élément suivant tant que l'on n'a pas atteint la fin du tableau.

```
Procédure TriInsertion(↔tab tableau[TMAX]
d'Entier, →nbElements : Entier)
Variable
  i, j: Entier
  bool : Booléen
Début
  Pour i de 1 à nbElements-1 faire
    bool ← vrai
    j←i
    TantQue (bool ET (j≥1))
      bool ← (tab[j-1] > tab[j])
      Si (bool) Alors
        swap(tab[j-1], tab[j])
      FinSi
      j ← j - 1
    FinTantQue
  FinPour
Fin
```

TRI PAR INSERTION

TRI RAPIDE

SAE S1.02

TRI RAPIDE (QUICKSORT)

PRINCIPE

La méthode consiste à placer un élément du tableau (appelé pivot) à sa place définitive, en permutant tous les éléments de telle sorte que tous ceux qui sont inférieurs au pivot soient à sa gauche et que tous ceux qui sont supérieurs au pivot soient à sa droite. Cette opération s'appelle le partitionnement.
Pour chacun des sous-tableaux, on définit un nouveau pivot et on répète l'opération de partitionnement. Ce processus est répété récursivement, jusqu'à ce que l'ensemble des éléments soit trié.
Le choix du pivot est un élément important de l'algorithme car il a un impact sur la complexité moyenne de l'algorithme.
Le pivot peut être :

- Fixé arbitrairement
- Fixé aléatoirement
- Déterminé par l'axe médian.

TRI RAPIDE (QUICKSORT)

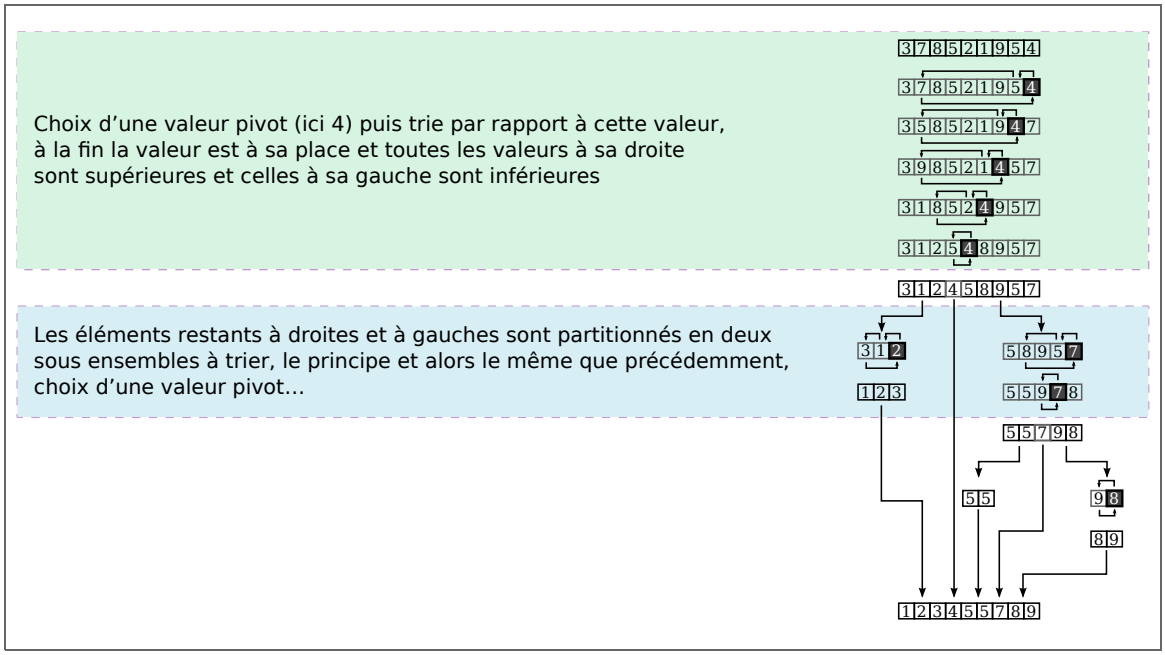


Illustration de l'algorithme de tri-rapide - source **wikipédia**

QUICKSORT

QUICKSORT

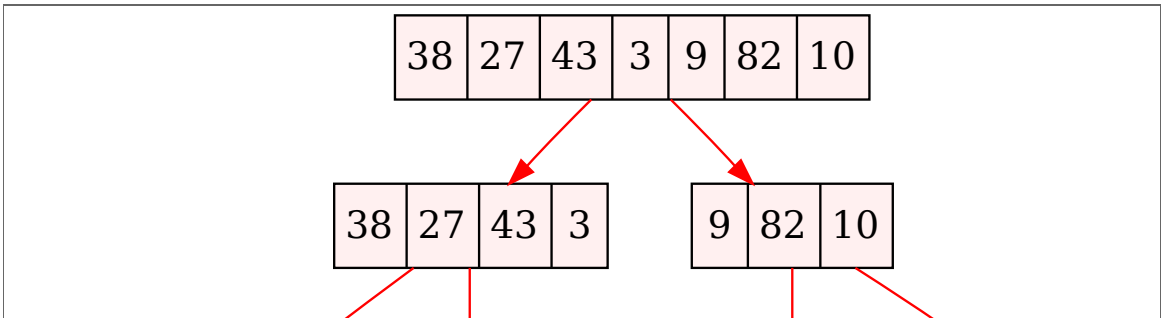
ALGORITHME

1. Choix d'un pivot
2. Décalage du pivot à la fin du tableau
3. Parcours des éléments `T[i]` de `T` avec $i \in [0..(n-2)]$
   Pour chaque élément, on teste si T[i] est inférieur au pivot
   - Si OUI alors
     - on permute `T[i]` et `T[k]`
     - On incrémente k et on passe à l'élément suivant
       k représente l'indice du premier élément qui était supérieur au pivot.
       Initialement $k = i = 0$.
   - Si NON, on passe directement à l'élément suivant.
4. Lorsque l'on a atteint la fin du tableau, on permute `T[i]` et `T[k]`
   Et on recommence pour chaque partition.

DIVISER POUR RÉGNER

PRINCIPE

Contrairement aux tris par sélection, par insertion ou encore par propagation (tir à bulles) qui utilisent une approche dite *naïve*, les tris rapide et introspectif (cf. **http://fr.wikipedia.org/wiki/Introsort**) utilisent une approche algorithmique appelée *diviser pour régner*, ou encore *divide and conquer*, en anglais.
Cette technique consiste à diviser le problème initial en sous-problèmes (diviser). Ensuite, on va résoudre chacun des sous-problèmes qui sont en principe plus simple à résoudre que le problème initial (régner). Enfin, on va combiner les solutions des sous-problèmes pour obtenir la solution du problème initial.
Ce type d'approche utilise souvent des algorithmes récursifs pour la résolution des sous-problèmes et permet d'exploiter le parallélisme ce qui les rend particulièrement efficace. Cependant, tous les problèmes ne se prêtent pas à cette technique.



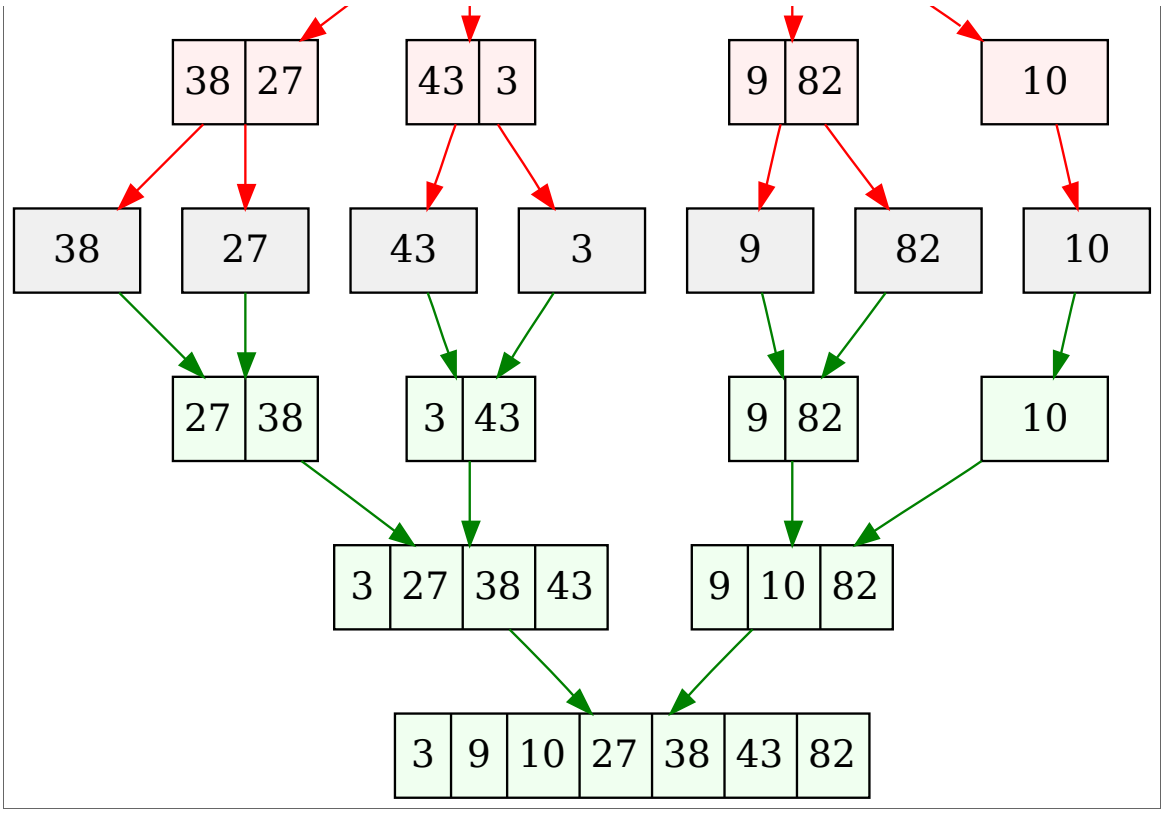


Illustration du tri fusion qui utilise une approche “diviser pour régner” - source **wikipédia**

SYNTHÈSE DES TRIS

SAE S1.02

COMPARAISON DES DIFFÉRENTS ALGORITHME DE TRI VUS PRÉCÉDEMMENT