Correction d'algorithmes de tri

Bloc 1 - DIU

Aix-Marseille Université

avril 2019

Tris.

organiser un ensemble d'objets selon un ordre déterminé

relation d'ordre : comparaison de clés

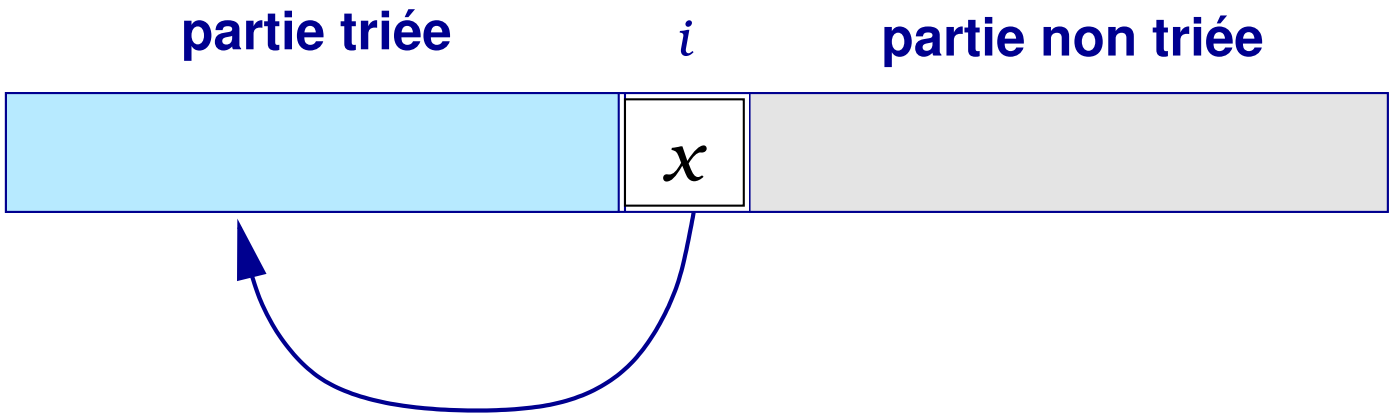
dans nos exemple nous confondrons les objets avec leurs clés

- tri par comparaison versus tri par indexation
- tri sur place : espace mémoire de taille constante
- tri stable : préserve l'ordre initial en cas d'égalité

Tris.

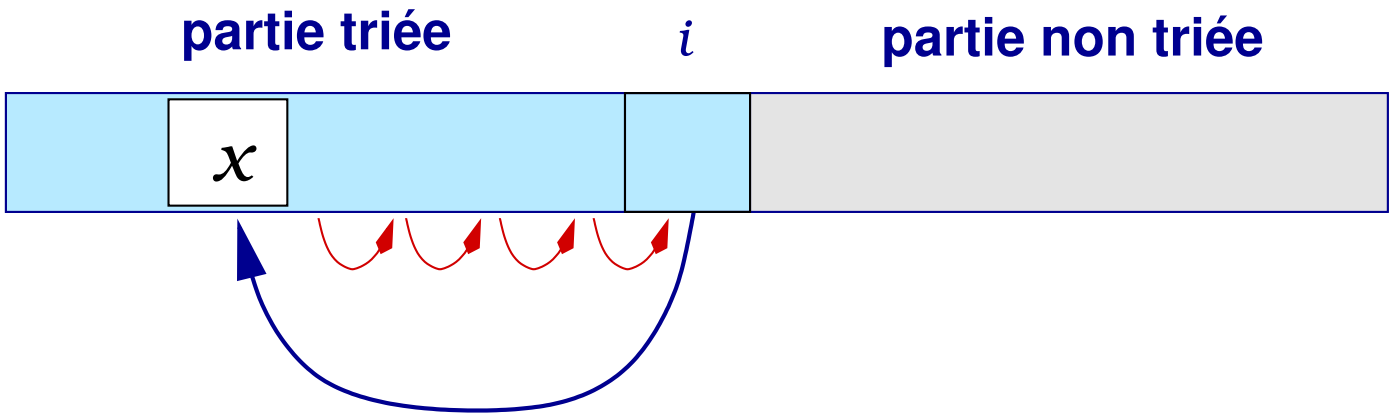
1. **Tris en $O(n^2)$.**
   - tri à bulles,
   - tri par insertion,
   - tri par sélection.
2. **Tris en $O(n \times \log\ n)$.**
   - tri par fusion,
   - tri par tas,
   - tri rapide (mais en $O(n^2)$ dans le pire des cas).

Tri par insertion



Principe : insérer les éléments les uns après les autres dans la partie triée

Tri par insertion



Principe : insérer les éléments les uns après les autres dans la partie triée

Insertion : recopie des éléments plus grands vers la droite

Tri par insertion

```
procédure TRI_PAR_INSERTION(T[1, ..., n])
début
    pour i := 2 jusqu'à n faire
        j := i,
        tant que ((j > 1) et (T[j] < T[j − 1])) faire
            // la suite T[1, ..., j − 1] est ordonnée
            PERMUTER(T, j, j − 1),
            j := j − 1,
        fin faire
        // la suite T[1, ..., i] est ordonnée
    fin faire
fin procédure
```

Tri rapide (quick sort)

fonction TRI_RAPIDE (T, g, d)

```
1    si (g < d) alors
2        | Choisir pivot dans T[g, ..., d],
3        | m :=PARTITIONNER(T, g, d, pivot),
4        | TRI_RAPIDE (T, g, m − 1),
5        | TRI_RAPIDE (T, m + 1, d),
```

- tri par comparaisons *en place*,
- tableau indexé,
- le plus rapide dans le cas général,
- très utilisé

Tri rapide (quick sort)

fonction TRI_RAPIDE (T, g, d)

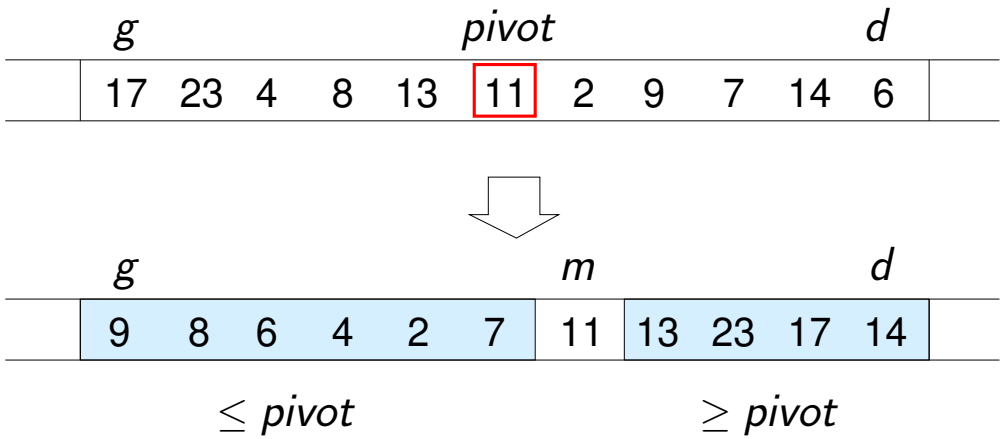
```
1   si (g < d) alors
2   |  Choisir pivot dans T[g, ..., d],
3   |  m :=PARTITIONNER(T, g, d, pivot),
4   |  TRI_RAPIDE (T, g, m − 1),          { m − 1 < d}
5   |  TRI_RAPIDE (T, m + 1, d),          { m + 1 > g}
```

Tri rapide (quick sort)

fonction TRI_RAPIDE (T, g, d)

```
1    si (g < d) alors
2        | Choisir pivot dans T[g, ..., d],
3        | m :=PARTITIONNER(T, g, d, pivot),
4        | TRI_RAPIDE (T, g, m − 1),          { m − 1 < d}
5        | TRI_RAPIDE (T, m + 1, d),          { m + 1 > g}
```

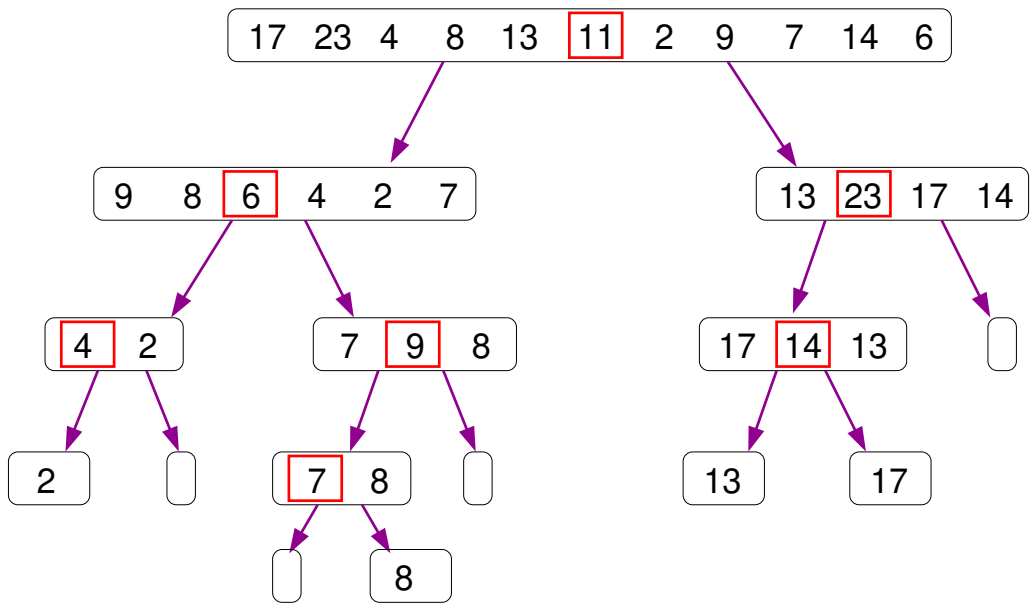
Partition :



Tri rapide (quick sort)

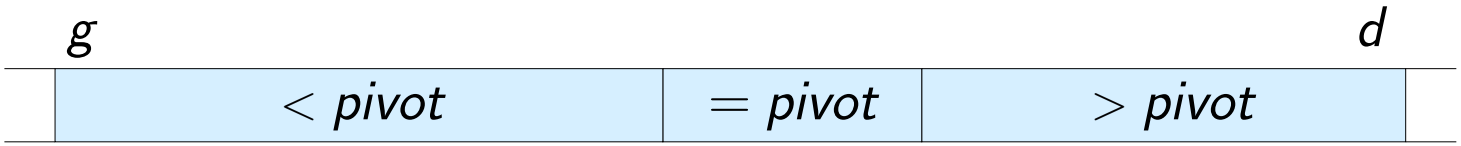
fonction TRI_RAPIDE (T, g, d)

```
1    si (g < d) alors
2        | Choisir pivot dans T[g, ..., d],
3        | m :=PARTITIONNER(T, g, d, pivot),
4        | TRI_RAPIDE (T, g, m − 1),
5        | TRI_RAPIDE (T, m + 1, d),
```



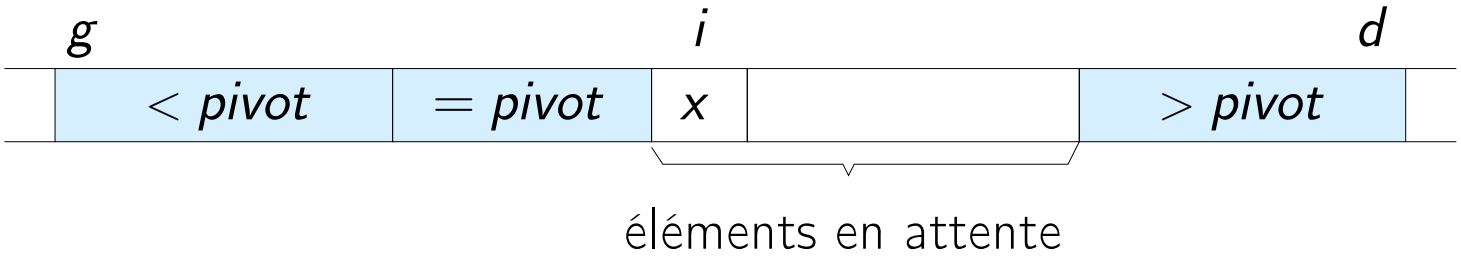
Partition type *drapeau*

Après la partition :



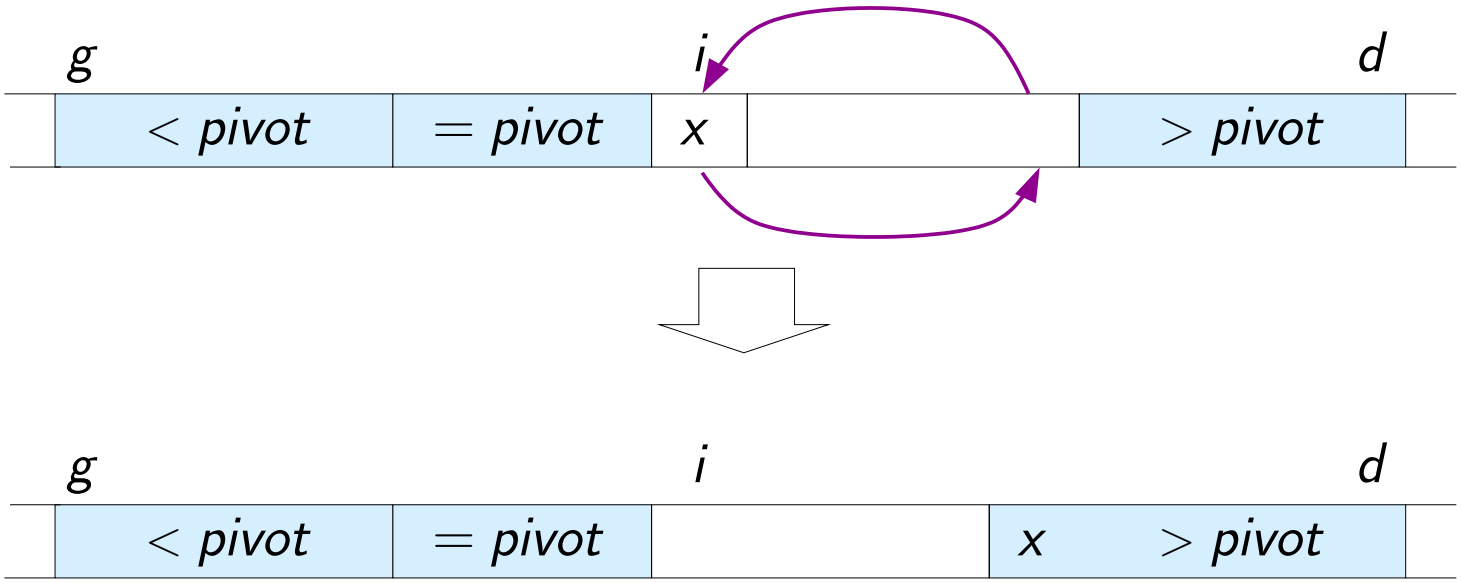
Partition type *drapeau*

Pendant la partition :



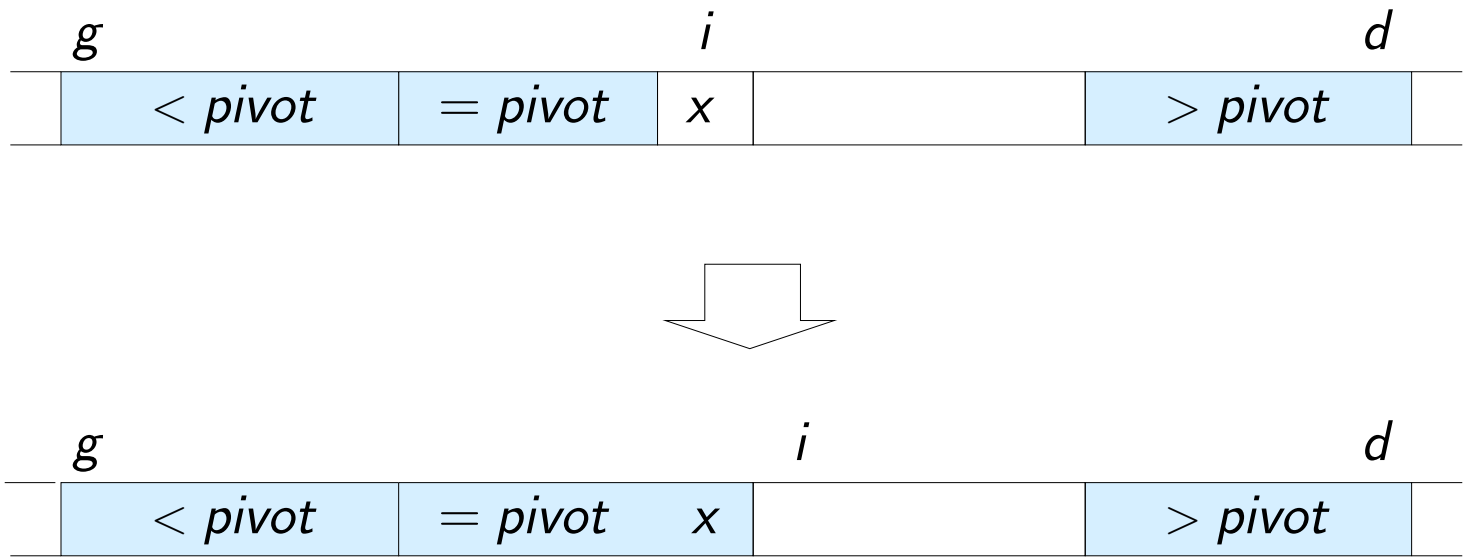
Partition type *drapeau*

cas 1 : $x > pivot$: permutation avec le dernier en attente



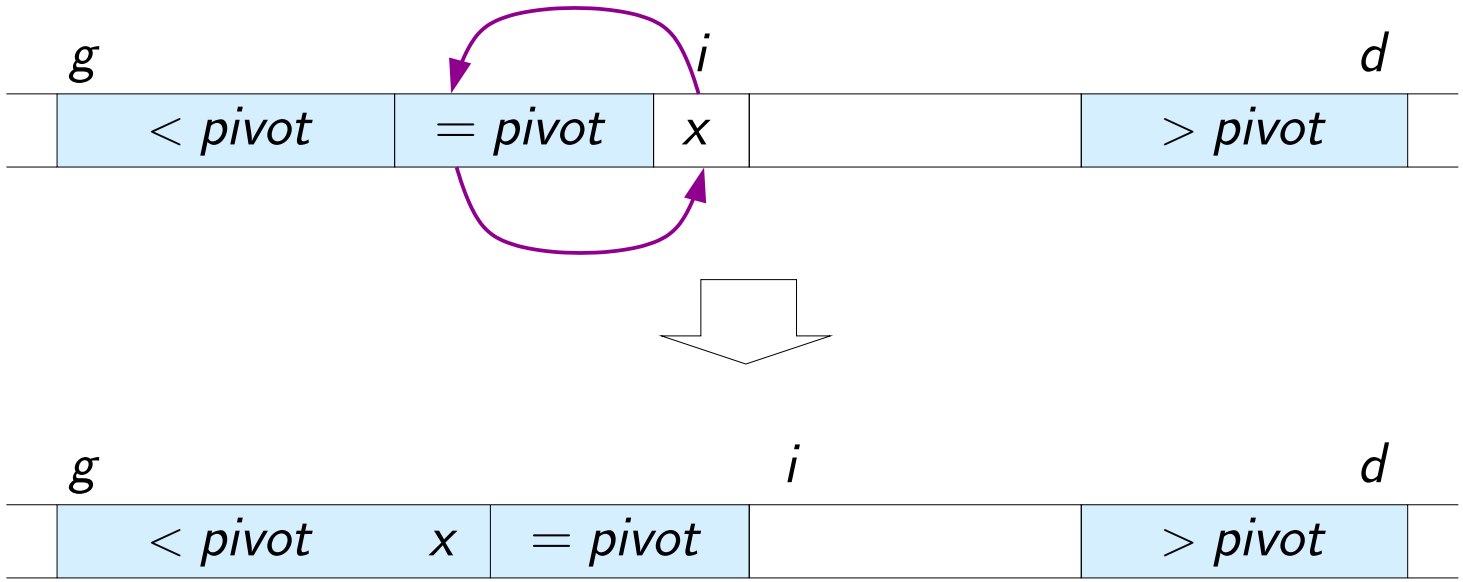
Partition type *drapeau*

cas 2 : $x = pivot$: extension de la zone



Partition type *drapeau*

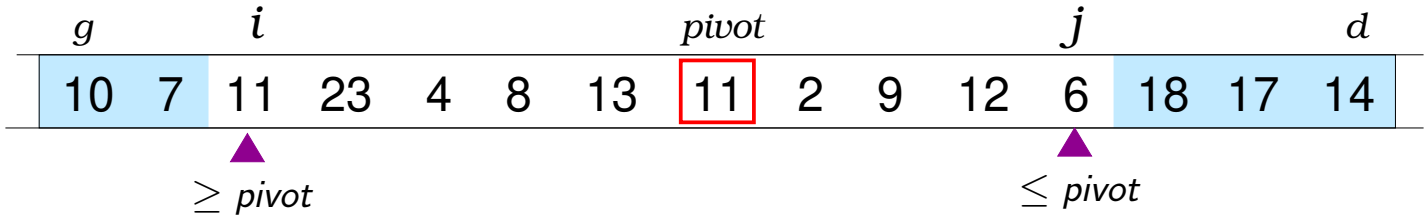
cas 3 : $x < pivot$: permutation avec le premier égal au pivot



Partition *rapide*

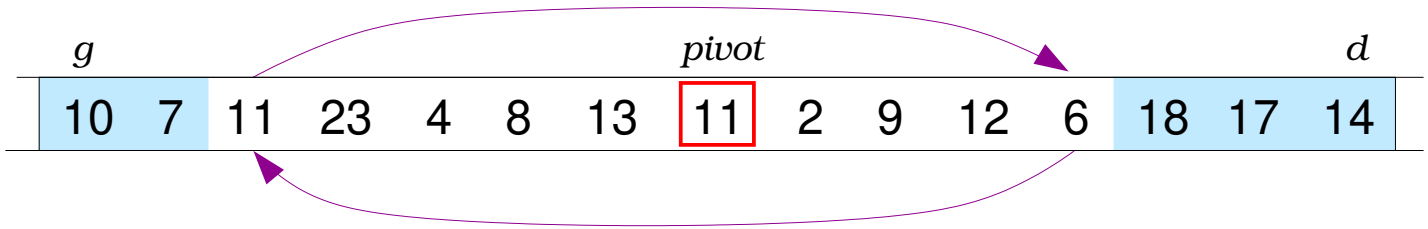
à gauche : stoppe avec $x_i \geq pivot$

à droite : stoppe avec $x_j \leq pivot$



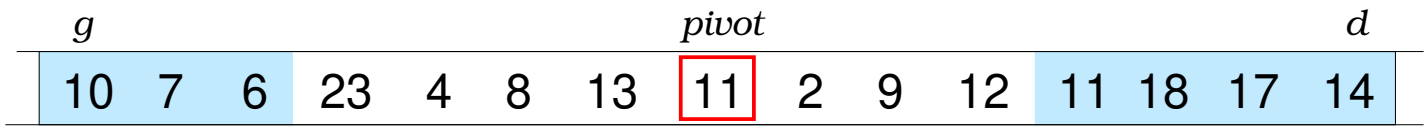
Partition *rapide*

Permutation



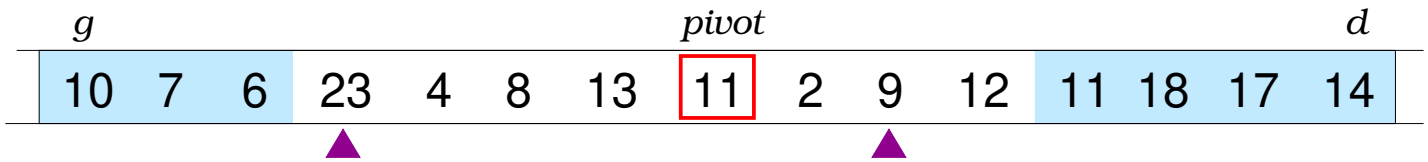
Partition *rapide*

Extension des zones



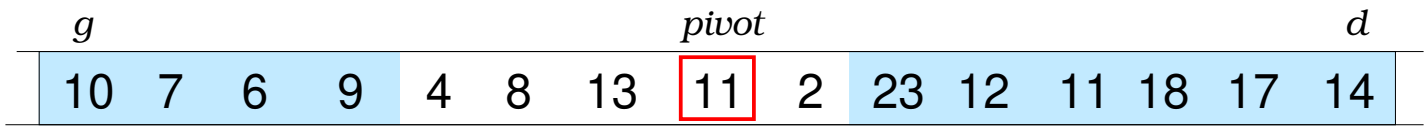
Partition *rapide*

Recherche d'un plus grand à gauche et d'un plus petit à droite



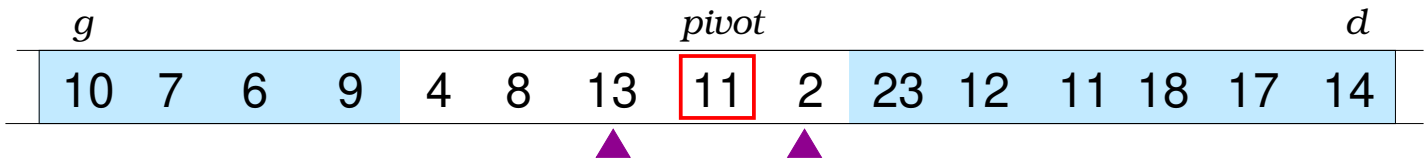
Partition *rapide*

Permutation et extension des zones



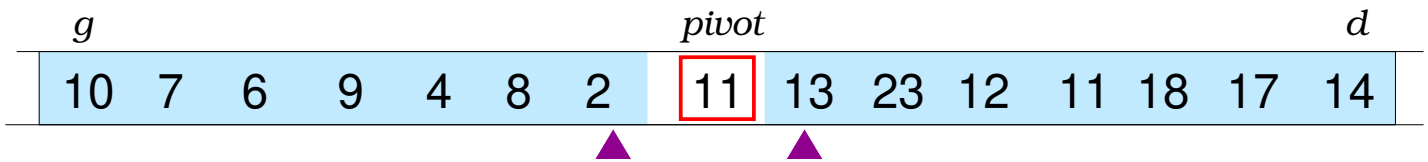
Partition *rapide*

Recherche d'un plus grand à gauche et d'un plus petit à droite



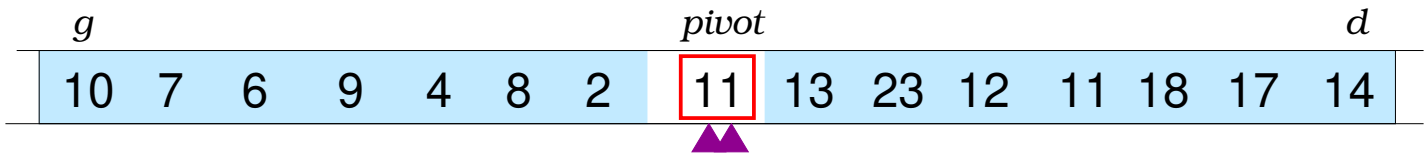
Partition *rapide*

Permutation et extension des zones



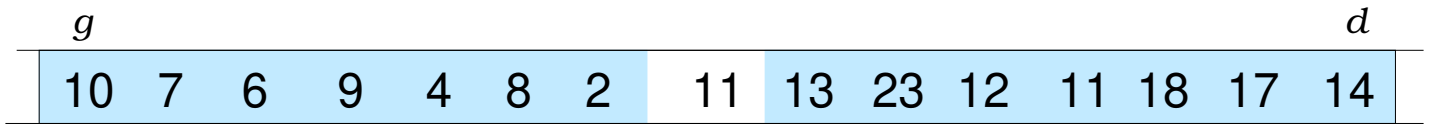
Partition *rapide*

Dernière permutation



Partition *rapide*

Produit deux zones strictement plus petites que $[g, d]$



La partition est en $\mathcal{O}(n)$

Quick sort avec partition rapide

fonction TRI_RAPIDE (T, g, d)

```
1   si (g < d) alors
2   |  a := g, b := d,
3   |  v := T[(a + b)/2],
4   |  tant que (a ≤ b) faire
5   |  |  tant que (T[a] < v) faire
6   |  |  |  a := a + 1,
7   |  |  tant que (T[b] > v) faire
8   |  |  |  b := b − 1,
9   |  |  si (a ≤ b) alors
10  |  |  |  PERMUTER(T, a, b),
11  |  |  |  a := a + 1,
12  |  |  |  b := b − 1,
13  |  TRI_RAPIDE (T, g, b),
14  |  TRI_RAPIDE (T, a, d),
15  fin fonction
```

Quick sort : terminaison

fonction TRI_RAPIDE (T, g, d)

```
1   si (g < d) alors                                  | n = d − g + 1 > 1
2       a := g, b := d,
3       v := T[(a + b)/2],
4       tant que (a ≤ b) faire
5           tant que (T[a] < v) faire                 | la première fois stoppée par le pivot,
6               a := a + 1,                           | ensuite stoppée par T[b]
7           tant que (T[b] > v) faire
8               b := b − 1,
9           si (a ≤ b) alors
10              PERMUTER(T, a, b),
11              a := a + 1,
12              b := b − 1,
13      TRI_RAPIDE (T, g, b),
14      TRI_RAPIDE (T, a, d),
15  fin fonction
```

Quick sort : terminaison

fonction TRI_RAPIDE (T, g, d)

```
1   si (g < d) alors                                  n = d − g + 1 > 1
2       a := g, b := d,
3       v := T[(a + b)/2],
4       tant que (a ≤ b) faire
5           tant que (T[a] < v) faire
6               a := a + 1,
7           tant que (T[b] > v) faire                 la première fois stoppée par le pivot,
8               b := b − 1,                           ensuite stoppée par T[a]
9           si (a ≤ b) alors
10              PERMUTER(T, a, b),
11              a := a + 1,
12              b := b − 1,
13      TRI_RAPIDE (T, g, b),
14      TRI_RAPIDE (T, a, d),
15  fin fonction
```

Quick sort : terminaison

fonction TRI_RAPIDE (T, g, d)

```
1   si (g < d) alors                                   n = d − g + 1 > 1
2       a := g, b := d,
3       v := T[(a + b)/2],
4       tant que (a ≤ b) faire
5           tant que (T[a] < v) faire
6               a := a + 1,
7           tant que (T[b] > v) faire
8               b := b − 1,
9           si (a ≤ b) alors                           forcément vrai la première fois
10              PERMUTER(T, a, b),
11              a := a + 1,
12              b := b − 1,
13      TRI_RAPIDE (T, g, b),
14      TRI_RAPIDE (T, a, d),
15  fin fonction
```

Quick sort : terminaison

fonction TRI_RAPIDE (T, g, d)

```
1   si (g < d) alors                                  n = d − g + 1 > 1
2       a := g, b := d,
3       v := T[(a + b)/2],
4       tant que (a ≤ b) faire
5           tant que (T[a] < v) faire
6               a := a + 1,
7           tant que (T[b] > v) faire
8               b := b − 1,
9           si (a ≤ b) alors                          forcément vrai la première fois
10              PERMUTER(T, a, b),
11              a := a + 1,                           on incrémente a au moins une fois
12              b := b − 1,                           on décrémente b au moins une fois
13      TRI_RAPIDE (T, g, b),
14      TRI_RAPIDE (T, a, d),
15  fin fonction
```

Quick sort : terminaison

fonction TRI_RAPIDE (T, g, d)

```
1   si (g < d) alors                                  | n = d − g + 1 > 1
2       a := g, b := d,
3       v := T[(a + b)/2],
4       tant que (a ≤ b) faire
5           tant que (T[a] < v) faire
6               a := a + 1,
7           tant que (T[b] > v) faire
8               b := b − 1,
9           si (a ≤ b) alors                          | forcément vrai la première fois
10              PERMUTER(T, a, b),
11              a := a + 1,                           | on incrémente a au moins une fois
12              b := b − 1,                           | on décrémente b au moins une fois
13      TRI_RAPIDE (T, g, b),                         | ici b < d donc b − g + 1 < n
14      TRI_RAPIDE (T, a, d),                         | ici a > g donc d − a + 1 < n
15  fin fonction
```

Quick sort : correction

fonction TRI_RAPIDE (T, g, d)

```
1   si (g < d) alors
2       a := g, b := d,
3       v := T[(a + b)/2],
4       tant que (a ≤ b) faire
            {∀i, si g ≤ i < a alors T[i] ≤ v, et si b < i ≤ d alors T[i] ≥ v,}
5           tant que (T[a] < v) faire
6               a := a + 1,                          {T[a − 1] < v}
7           tant que (T[b] > v) faire
8               b := b − 1,                          {T[b + 1] > v}
9           si (a ≤ b) alors
10              PERMUTER(T, a, b),
                {T[a] ≤ v et T[b] ≥ v}
11              a := a + 1,
12              b := b − 1,
            {donc si i < a alors T[i] ≤ v et si i > b alors T[i] ≥ v,}
13      TRI_RAPIDE (T, g, b),                        {b < a et ∀i, i < a on a T[i] ≤ v}
14      TRI_RAPIDE (T, a, d),                        {a > b et ∀i, i > b on a T[i] ≥ v}
15  fin fonction
```