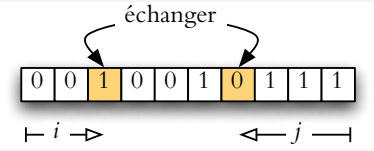
13 Tri rapide

13.1 Tri binaire

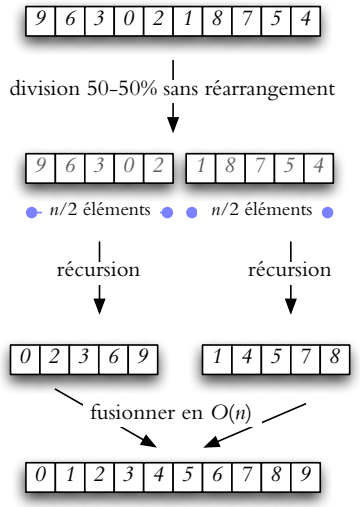
Supposons qu'il y a juste deux clés possibles (0 et 1) dans un tableau à trier. Alors on peut performer le tri en un temps linéaire à l'aide de deux indices qui balayent à partir des extremités vers le milieu.



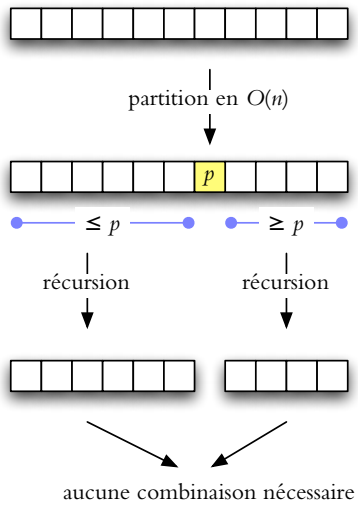

```
    TRI01(A[0..n − 1])                                   // tri binaire
B1  i ← 0 ; j ← n − 1
B2  loop
B3      while A[i] = 0 et i < j do i ← i + 1
B4      while A[j] = 1 et i < j do j ← j − 1
B5      if i < j then échanger A[i] ↔ A[j]
B6      else return
```

13.2 Tri rapide

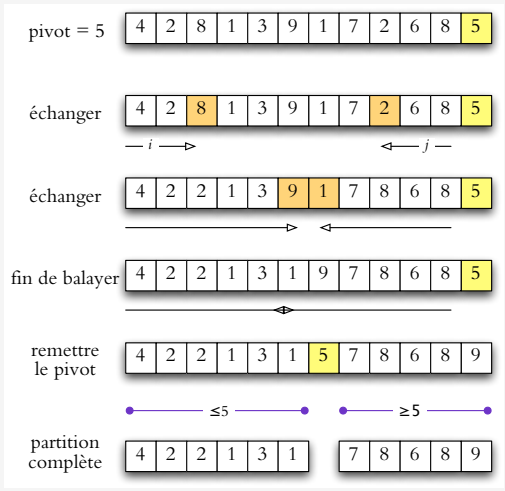
W(fr)



Le **tri par fusion** utilise la logique de «diviser pour régner» : le tableau est divisé en deux sous-tableaux (en temps $O(1)$) qui sont triés ensuite dans des appels récursifs, et on combine les résultats (fusion) en un temps linéaire.



En **tri rapide**, on choisit un **pivot** p, et à l'aide des échanges d'éléments, on place les éléments inférieurs à p à la gauche, et ceux supérieurs à p à la droite du tableau. Après une telle partition, on peut procéder avec des appels récursifs aux deux sous-tableaux gauche et droit. Notez que le pivot n'est pas nécessairement la médiane : les sous-tableaux gaches et droits résultants peuvent avoir des tailles très différentes. La partition même suit la logique du tri binaire.



L'idée principale est la **partition** autour d'un pivot.

Algo QUICKSORT$(A[0..n-1], g, d)$ *// tri de $A[g..d]$*
Q1 **if** $g \geq d$ **then return** *// cas de base*
Q2 $i \leftarrow$ PARTITION(A, g, d)
Q3 QUICKSORT$(A, g, i-1)$
Q4 QUICKSORT$(A, i+1, d)$

Algo PARTITION(A, g, d) *// partition de $A[g..d]$*
P1 chosir le pivot $p \leftarrow A[d]$
P2 $i \leftarrow g-1$; $j \leftarrow d$
P3 **loop**
P4 **do** $i \leftarrow i+1$ **while** $A[i] < p$
P5 **do** $j \leftarrow j-1$ **while** $j \geq i$ et $A[j] > p$
P6 **if** $i \geq j$ **then** sortir de la boucle
P7 échanger $A[i] \leftrightarrow A[j]$
P8 échanger $A[i] \leftrightarrow A[d]$
P9 **return** i

Pour trier un tableau $A[0..n-1]$ en ordre croissant, on exécute QUICKSORT$(A, 0, n-1)$. C'est un tri en place.

13.3 Performances

Soit $m = d - g + 1$, le nombre des éléments dans le sous-tableau à trier, avec $m > 1$. La partition (Lignes P3–P7) se fait en un temps $\Theta(m)$. Le temps de calcul est donc

$$T(m) = \Theta(m) + T(i) + T(m-1-i).$$

La récurrence dépend de l'indice i du pivot.

	pivot i	**récurrence** $T(n)$	**solution** $T(n)$
Meilleur cas	$(n-1)/2$	$2 \cdot T\big((n-1)/2\big) + \Theta(n)$	$\Theta(n \log n)$
Pire cas	$0, n-1$	$T(n-1) + \Theta(n)$	$\Theta(n^2)$
Moyen cas	aléatoire	$\mathbb{E}T(n) = 2\mathbb{E}T(i) + \Theta(n)$	$\Theta(n \log n)$

Le pire cas arrive (entre autres) quand on a un tableau trié au début !