Les Tris par comparaison

Yann Strozecki
yann.strozecki@uvsq.fr

Octobre 2016

Questions courantes

- Comment calculer la médiane ou les quantiles d'un ensemble de nombres ?
- Construction d'un dictionnaire ou d'un annuaire ?

Questions courantes

- Comment calculer la médiane ou les quantiles d'un ensemble de nombres ?
- Construction d'un dictionnaire ou d'un annuaire ?
- Comment éliminer les doublons dans un tableau ?

Questions courantes

- Comment calculer la médiane ou les quantiles d'un ensemble de nombres ?
- Construction d'un dictionnaire ou d'un annuaire ?
- Comment éliminer les doublons dans un tableau ?
- Trouver la différence minimum entre deux entiers quelconque dans un tableau.

Questions courantes

- Comment calculer la médiane ou les quantiles d'un ensemble de nombres ?
- Construction d'un dictionnaire ou d'un annuaire ?
- Comment éliminer les doublons dans un tableau ?
- Trouver la différence minimum entre deux entiers quelconque dans un tableau.

Le tri c'est à dire ordonner les éléments d'un structure linéaire (tableau, liste) en ordre croissant.

Questions courantes

- Comment calculer la médiane ou les quantiles d'un ensemble de nombres ?
- Construction d'un dictionnaire ou d'un annuaire ?
- Comment éliminer les doublons dans un tableau ?
- Trouver la différence minimum entre deux entiers quelconque dans un tableau.

Le tri c'est à dire ordonner les éléments d'un structure linéaire (tableau, liste) en ordre croissant.

Les tris par comparaison

Données

- Collection de $TailleMax$ valeurs du même type rangées dans un tableau T
- Un opérateur de comparaison implémentant un ordre, par exemple $\leq$

But

Ré-ordonner les valeurs de T de telle sorte que :

$$T[i] \leq T[i+1], \forall i \in \{1 \ldots TailleMax - 1\}$$

Quelques algorithmes de tris

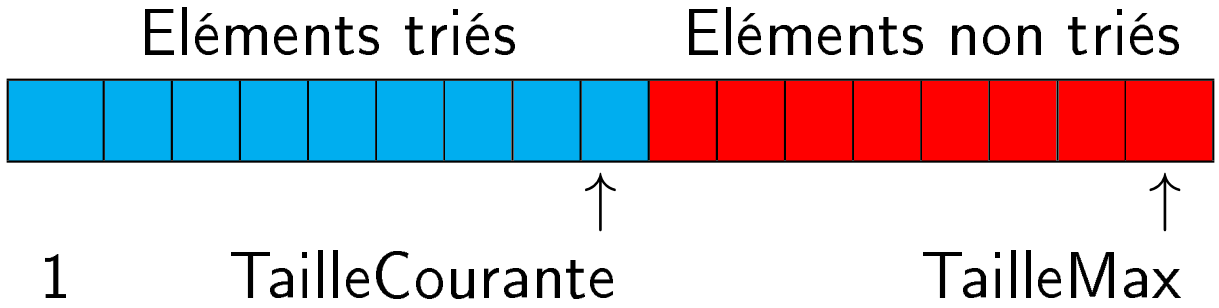
- Le tri par insertion
- Le tri par sélection
- Le tri à bulles (par permutation)
- Le tri fusion
- Le tri rapide (Quicksort)

Le tri par insertion

Principe Général

A tout moment le tableau T est séparé en 2 parties :

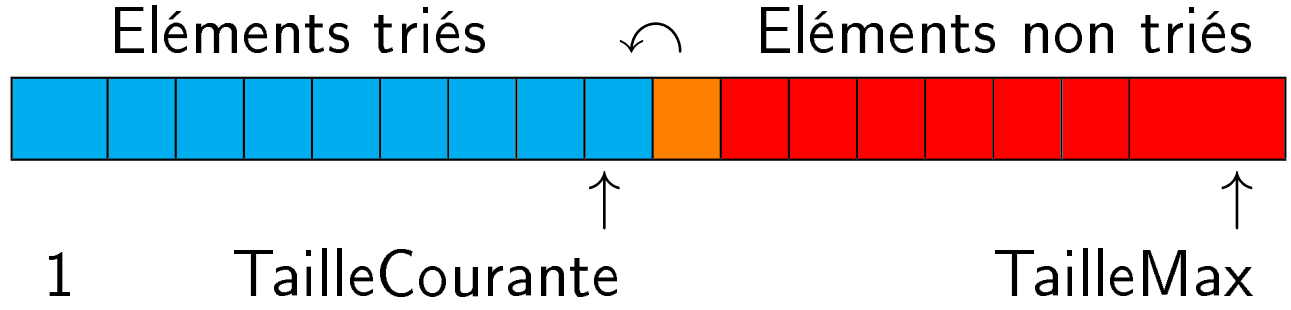
- $T[1] \ldots T[TailleCourante]$: Partie déjà triée du tableau
- $T[TailleCourante + 1] \ldots T[TailleMax]$: Partie non triée du tableau.



Le tri par insertion (2)

Une Etape

- Prendre un élément non encore trié ;
- L'insérer à sa place dans l'ensemble des éléments triés.



Le tri par insertion (3)

Algorithme 1 Tri par insertion

TriInsertion(T : **tableau d'entiers**, $TailleMax$: **entier**)

▷ ***Variables Locales***

$TC, i, p, temp$: **entiers**

Début

pour TC **de 1 à TailleMax - 1 faire**

$temp \leftarrow T[TC+1]$

$p \leftarrow 1$

tant que $T[p] < temp$ **faire**
p $\leftarrow$ **p** $+$ **1**
fin tant que

Chercher la position p

pour i de TC en décroissant à p faire
T[i+1] $\leftarrow$ **T[i]**
fin pour

Décaler les éléments

T[p] $\leftarrow$ **temp**

fin pour

Fin

Le tri par insertion (4)

Complexité pour n éléments

- Le corps de la boucle est exécuté $n-1$ fois
- Une itération :
  - Recherche de la position : p
  - Décalage des éléments : $TC - p$
  - Total : TC
- Au total :

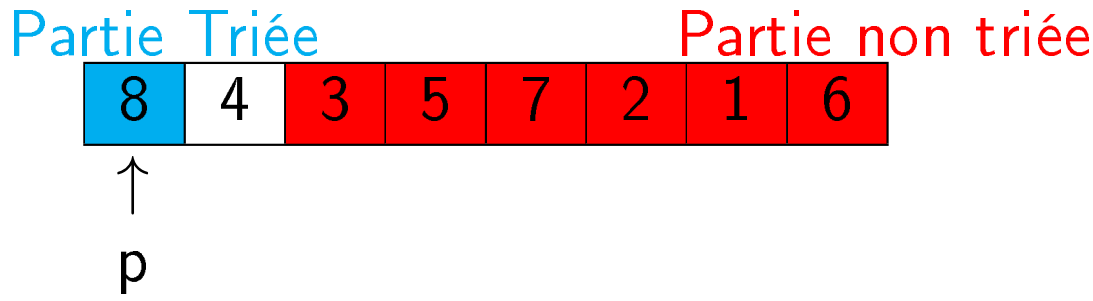
$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

La complexité du tri par insertion est en $O(n^2)$.

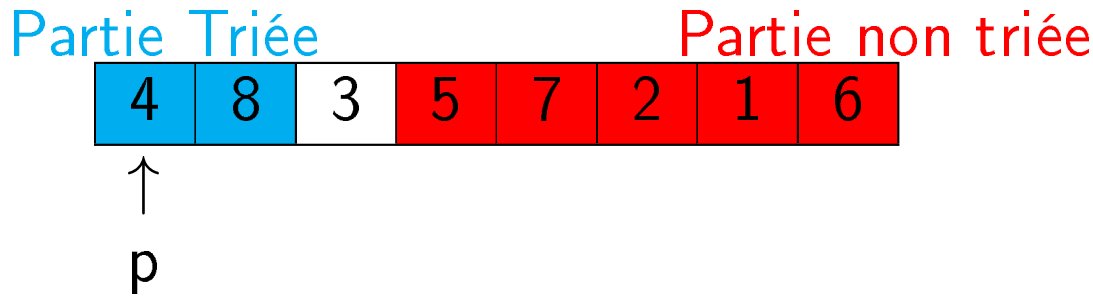
Un exemple de tri par insertion

8	4	3	5	7	2	1	6

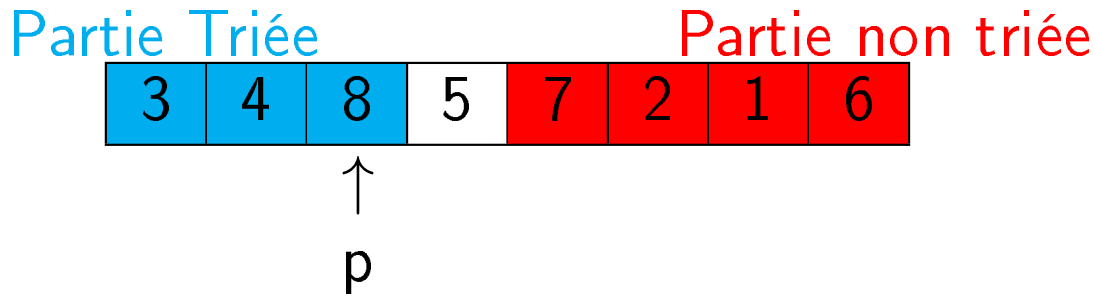
Un exemple de tri par insertion



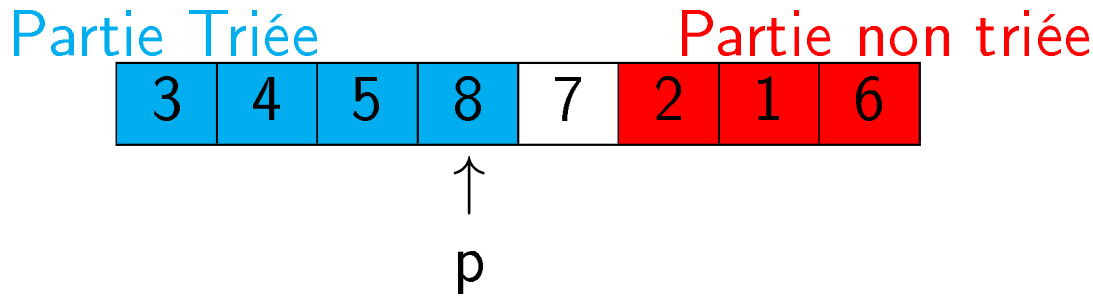
Un exemple de tri par insertion



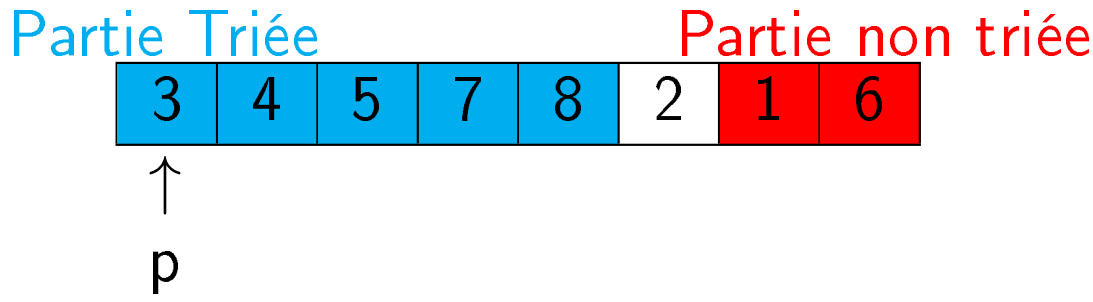
Un exemple de tri par insertion



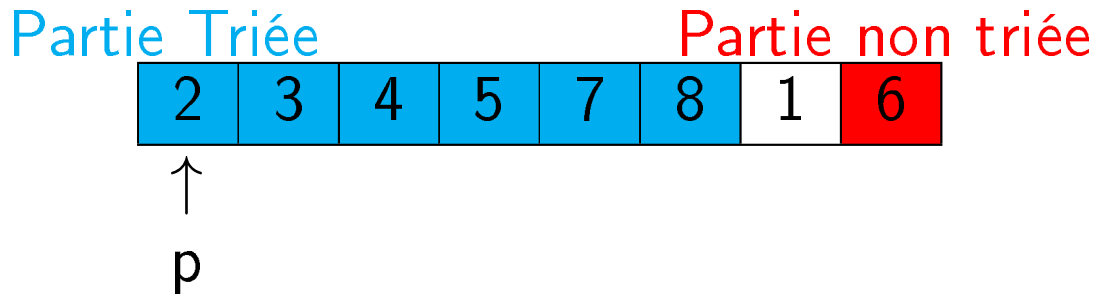
Un exemple de tri par insertion



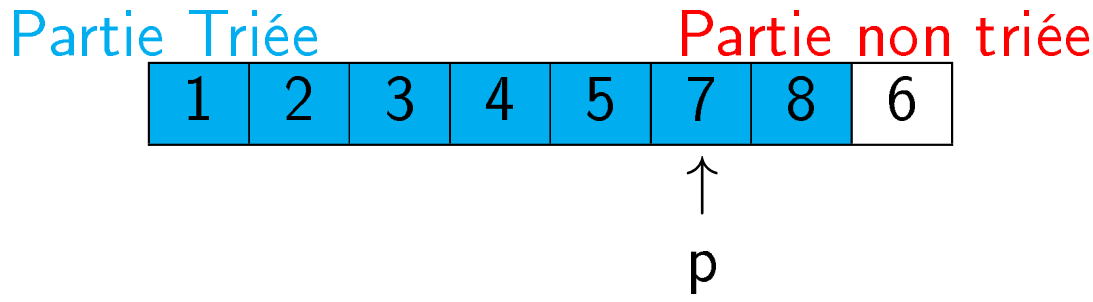
Un exemple de tri par insertion



Un exemple de tri par insertion



Un exemple de tri par insertion

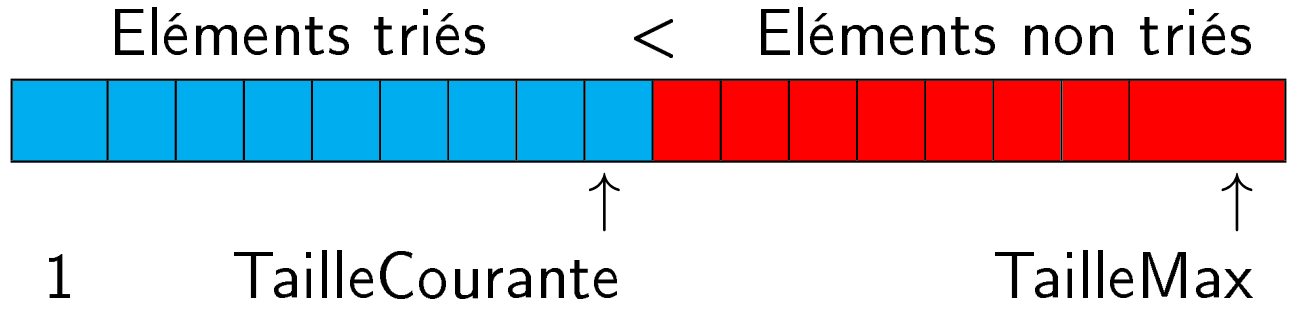


Un exemple de tri par insertion

Le tri par permutation (tri à bulles)

Principe général

- Si deux éléments voisins ne sont pas ordonnés correctement, on les échange.
- Deux parties dans le tableau :
  - Une partie avec des éléments triés
  - Une partie avec des éléments non triés
- Les éléments de la partie triée sont inférieurs aux éléments de la partie non triée.



Tri par permutation (2)

Algorithme 2 Tri par permutation

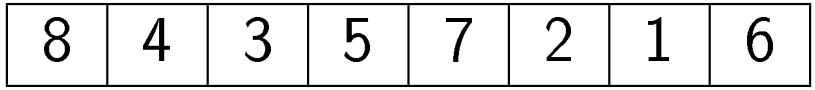
```
TriPermutation(T : tableau d'entiers, TailleMax : entier)
▷ Variables Locales
    i, TC : entiers
Début
pour TC de 2 à TailleMax faire
    pour i de TailleMax en décroissant à TC faire
        si T[i-1] > T[i] faire
            T[i-1] ↔ T[i]
        fin si

    fin pour

fin pour
Fin
```

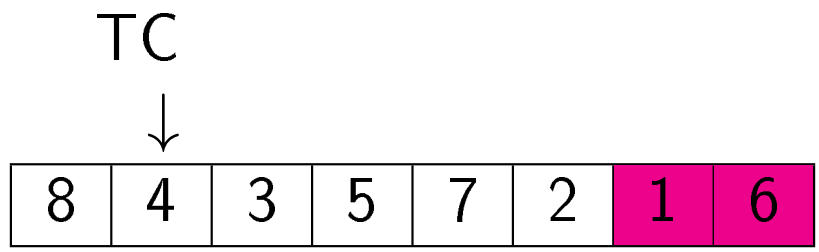
Tri par permutation (3)

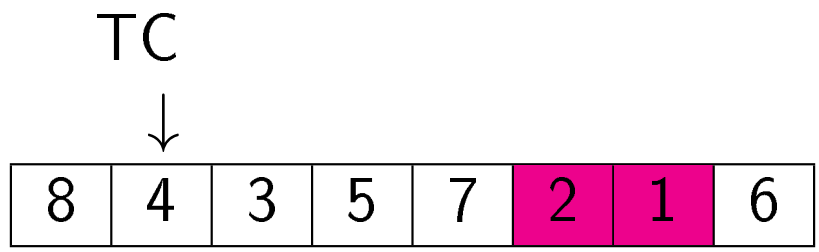
Complexité pour n éléments

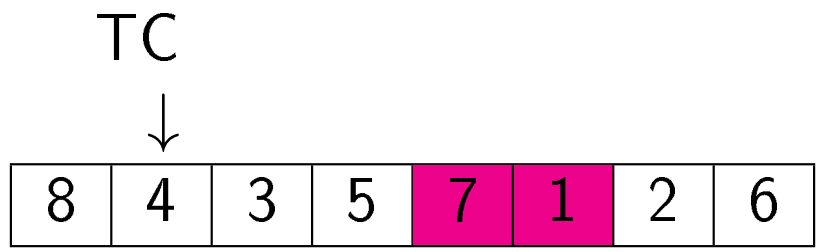
- Boucle externe : $n - 2$ fois
- Boucle interne : $TailleMax - TC$ fois
- Total : $\frac{(n-1)(n-2)}{2}$

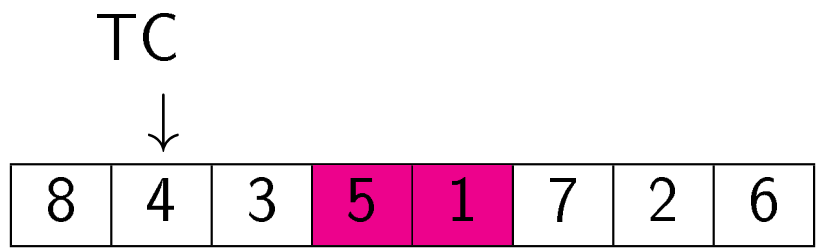
La complexité du tri par permutation est en $O(n^2)$.

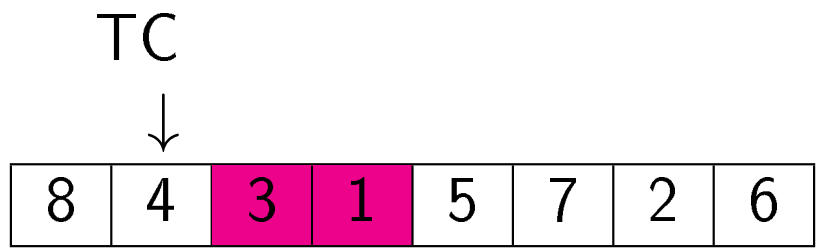

8 4 3 5 7 2 1 6

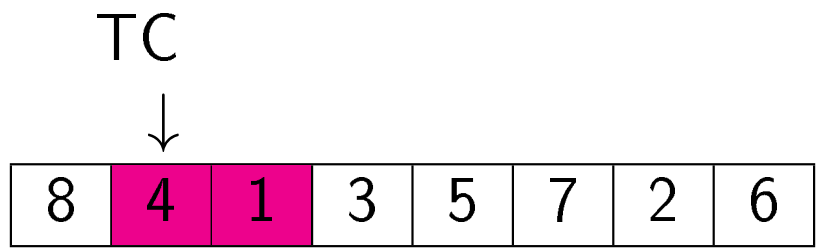

TC
8
4
3
5
7
2
1
6

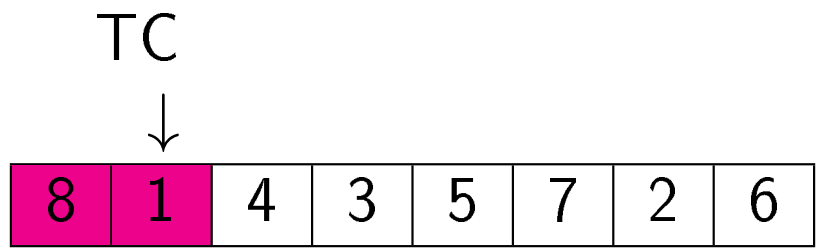

TC
8
4
3
5
7
2
1
6

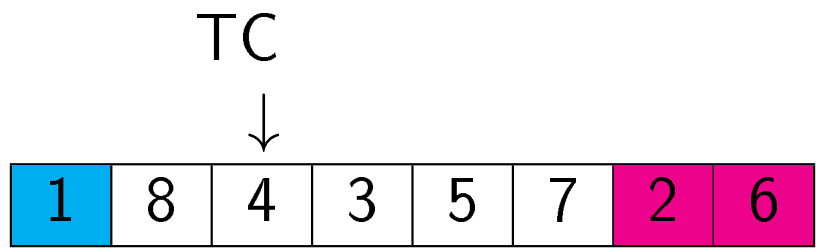

TC
8
4
3
5
7
1
2
6

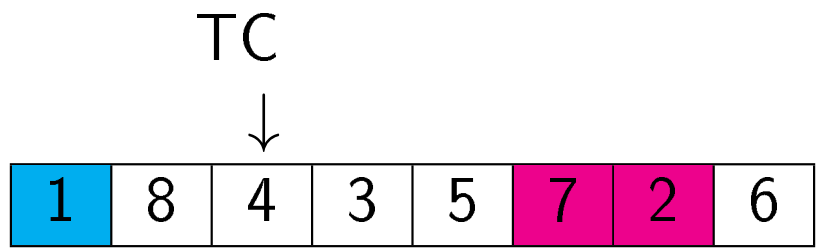

TC
8
4
3
5
1
7
2
6

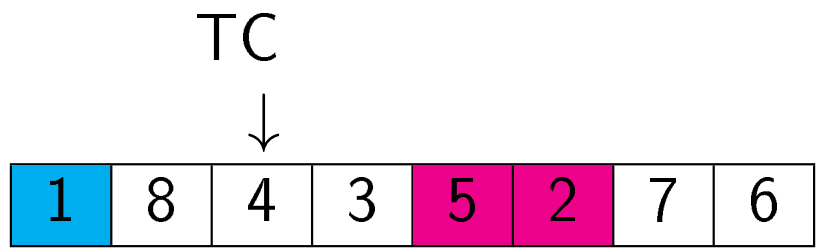

TC
8
4
3
1
5
7
2
6

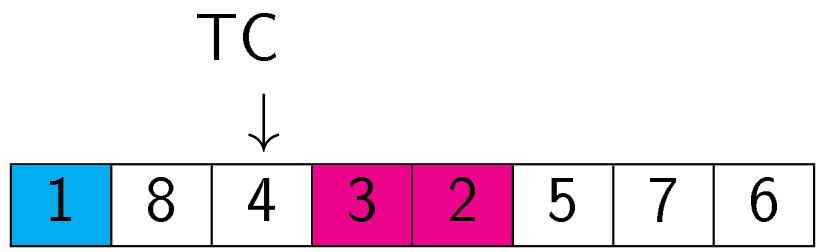

TC
8
4
1
3
5
7
2
6

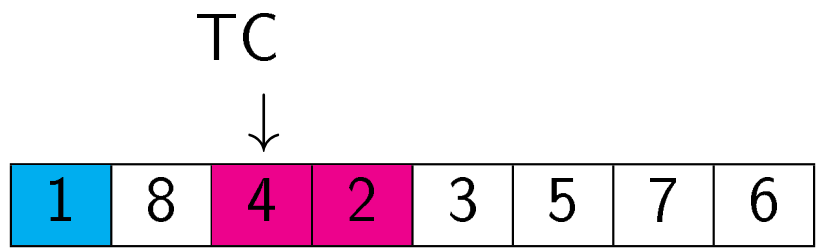

TC
8
1
4
3
5
7
2
6

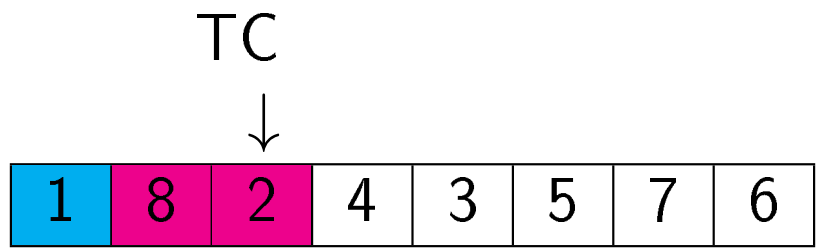

TC
1
8
4
3
5
7
2
6

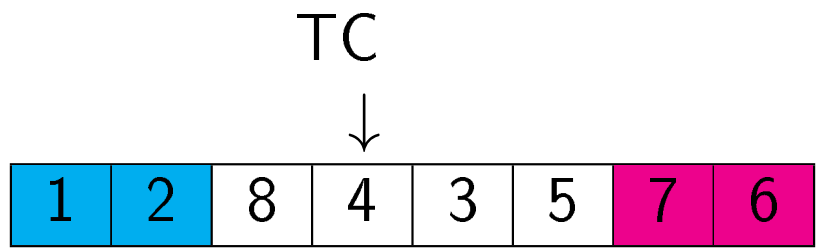

TC
1
8
4
3
5
7
2
6

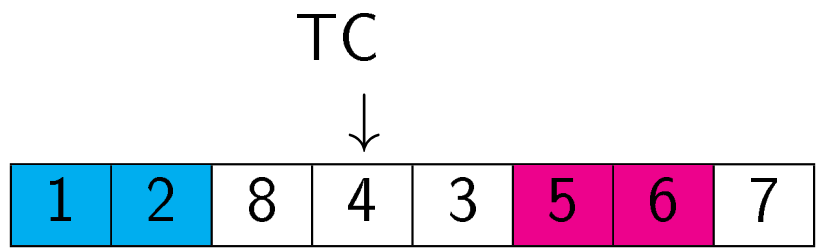

TC
1
8
4
3
5
2
7
6

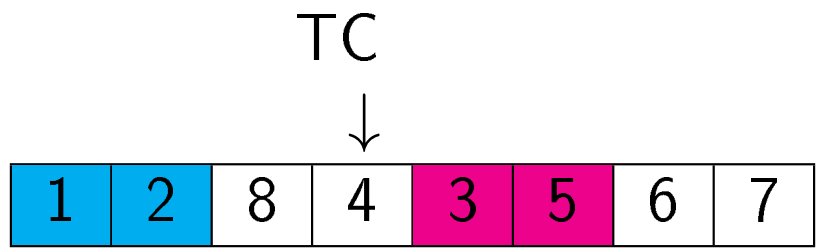

TC
1
8
4
3
2
5
7
6

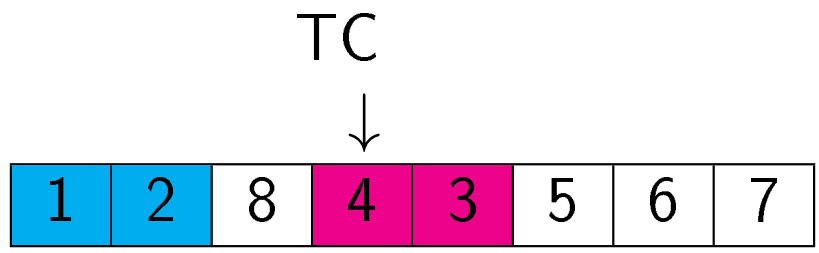

TC
1
8
4
2
3
5
7
6

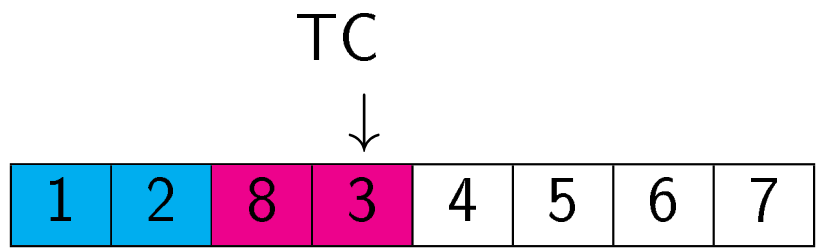

TC
1
8
2
4
3
5
7
6

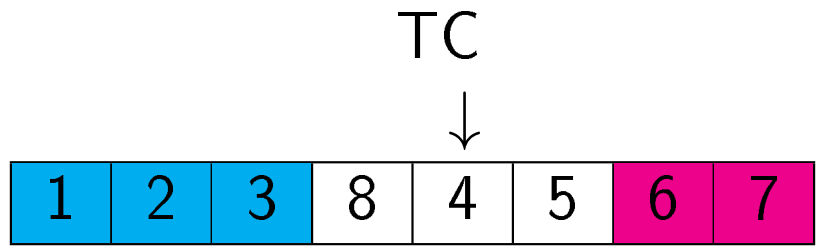

TC
1
2
8
4
3
5
7
6

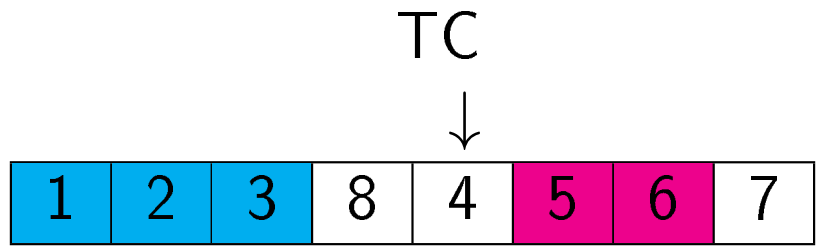

TC
1
2
8
4
3
5
6
7

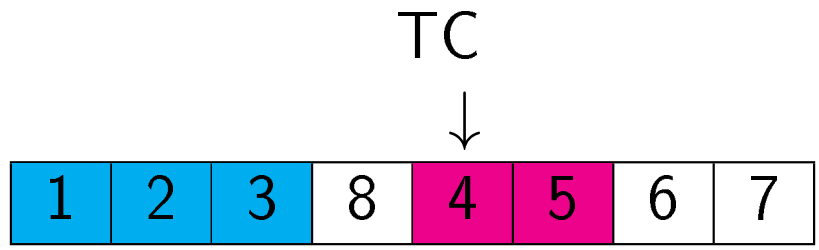

TC
1
2
8
4
3
5
6
7

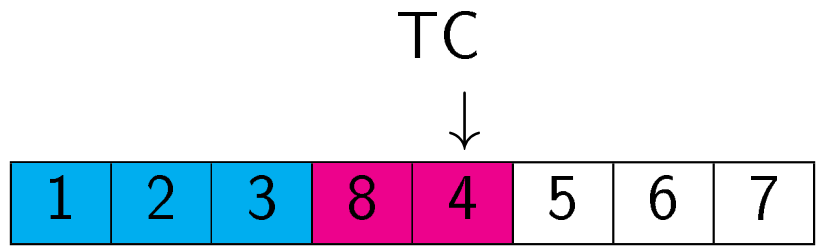

TC
1
2
8
4
3
5
6
7

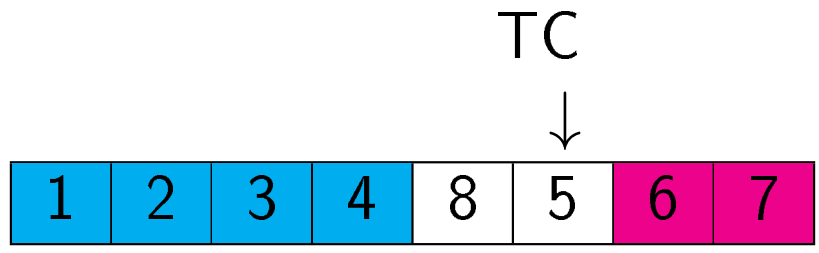

TC
1
2
8
3
4
5
6
7

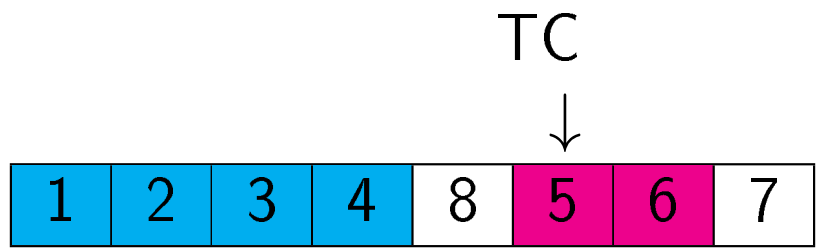

TC
1
2
3
8
4
5
6
7

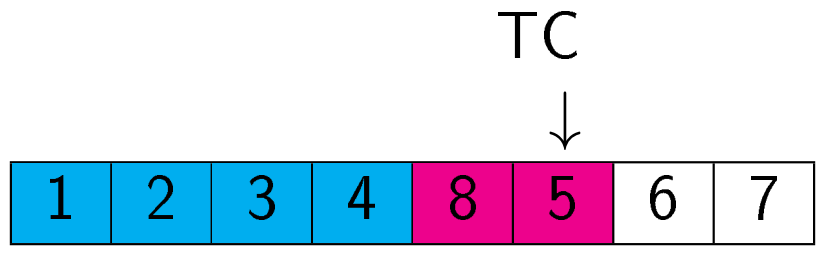

TC
1
2
3
8
4
5
6
7

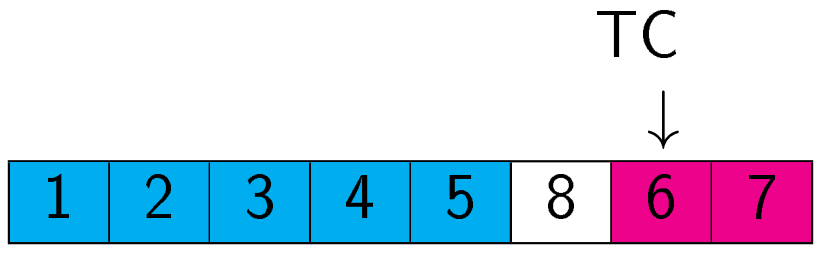

TC
1
2
3
8
4
5
6
7

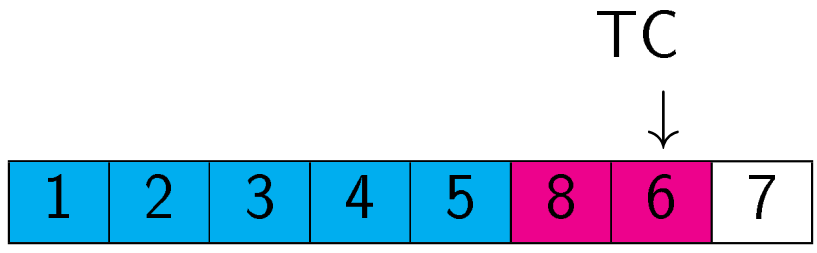

TC
1
2
3
8
4
5
6
7

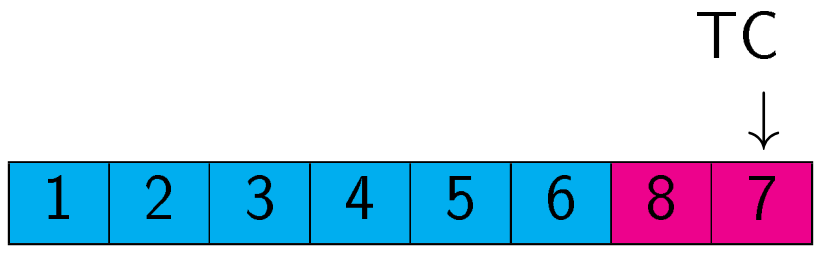

TC
1
2
3
4
8
5
6
7

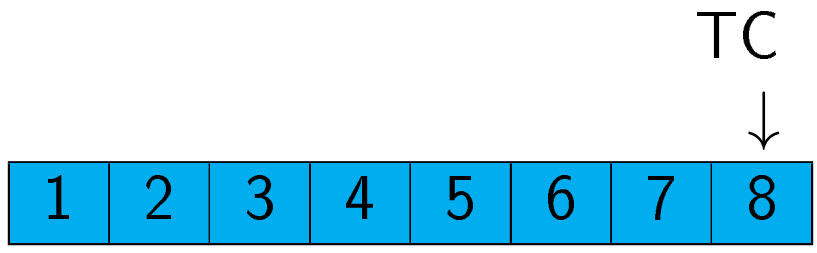

TC
1
2
3
4
8
5
6
7


TC
1
2
3
4
8
5
6
7


TC
1
2
3
4
5
8
6
7


TC
1
2
3
4
5
8
6
7


TC
1
2
3
4
5
6
8
7


TC
1
2
3
4
5
6
7
8

Complexité minimale pour un tri

On a donné deux algorithmes simples pour le tri mais peut on en trouver avec une meilleure complexité ?

Les bornes inférieures de complexité sont dures à prouver et requièrent un modèle précis.

Complexité minimale pour un tri

On a donné deux algorithmes simples pour le tri mais peut on en trouver avec une meilleure complexité ?

Les bornes inférieures de complexité sont dures à prouver et requièrent un modèle précis.

Ici on suppose qu'on sait comparer deux objets en temps constant mais on ne sait pas faire d'autre opération sur les objets à trier.

Complexité minimale pour un tri

On a donné deux algorithmes simples pour le tri mais peut on en trouver avec une meilleure complexité ?

Les bornes inférieures de complexité sont dures à prouver et requièrent un modèle précis.

Ici on suppose qu'on sait comparer deux objets en temps constant mais on ne sait pas faire d'autre opération sur les objets à trier.

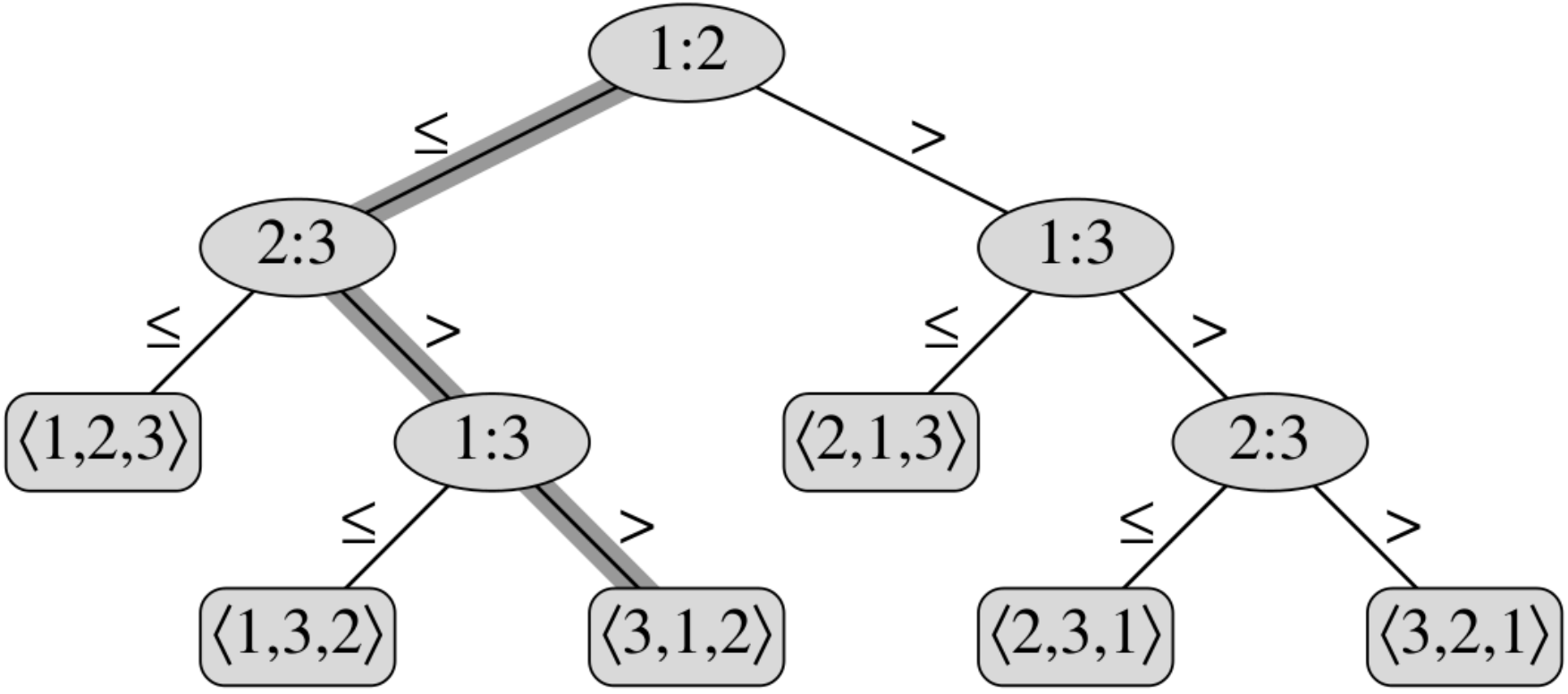
Arbre de comparaison

Le fonctionnement d'un algorithme de tri par comparaison sur toutes les entrées de taille n fixée peut être décrit par un arbre de comparaison.

- Ses noeuds internes sont étiquetés par des pairs d'entiers (i, j)
- Ses feuilles sont étiquetées par des permutations
- Ses arêtes gauches sont étiquetées par $\leq$ et celles de droite par $>$

Un noeud (i, j) correspond à la comparaison de l'élément i avec l'élément j. On suit l'arête en fonction du résultat de cette comparaison.
On abstrait les opérations d'écriture dans le tableau.

Illustration tirée du Cormen



Borne inférieure de complexité

Theorem

Tout tri par comparaison requiert au moins $\Omega(n \log n)$ comparaisons.

La démonstration se base sur les trois faits suivants :

- le nombre de permutations de taille n est $n!$
- un arbre binaire de hauteur h a au plus 2^h feuilles
- la formule de Stirling : $\log(n!) = O(n \log n)$

Peut-on trouver un tri dont la complexité est $O(n \log n)$?

Borne inférieure de complexité

Theorem

Tout tri par comparaison requiert au moins $\Omega(n \log n)$ comparaisons.

La démonstration se base sur les trois faits suivants :

- le nombre de permutations de taille n est $n!$
- un arbre binaire de hauteur h a au plus 2^h feuilles
- la formule de Stirling : $\log(n!) = O(n \log n)$

Peut-on trouver un tri dont la complexité est $O(n \log n)$?