Algorithmique E3

Algorithmes de tri

Daniel Vaz

daniel.ramosvaz@esiee.fr

2025-12-08

Notation

et	$\wedge$	&&
ou	$\vee$	||
not	$\neg$	!
égal	=	==
affectation	$\leftarrow$	=

$a \wedge b \leftrightarrow a$ et b

---

for $0 \leq i < n$

```
for (int i=0; i<n; i++)
    printf("%d\n", t[i]);
```

$t[a : b]$

éléments de indice a à $b-1$

Annonces

TD2 et TP4 cette semaine

- Algorithmes de tri

Exercices PLaTon

- Ont été réinitialisés en raison de la correction d'un bug
- Pas besoin de répéter, car ils ne sont pas évalués
- Pensez à conserver vos réponses avec vos fichiers TP

Annonces

TD2 et TP4 cette semaine

- Algorithmes de tri

Exercices PLaTon

- Ont été réinitialisés en raison de la correction d'un bug
- Pas besoin de répéter, car ils ne sont pas évalués
- Pensez à conserver vos réponses avec vos fichiers TP

Évaluation

- note de l'examen écrit
- exercices de écrite de code en pseudo-code
- exercices d'analyse de code C (QCM)

Aperçu

Révision

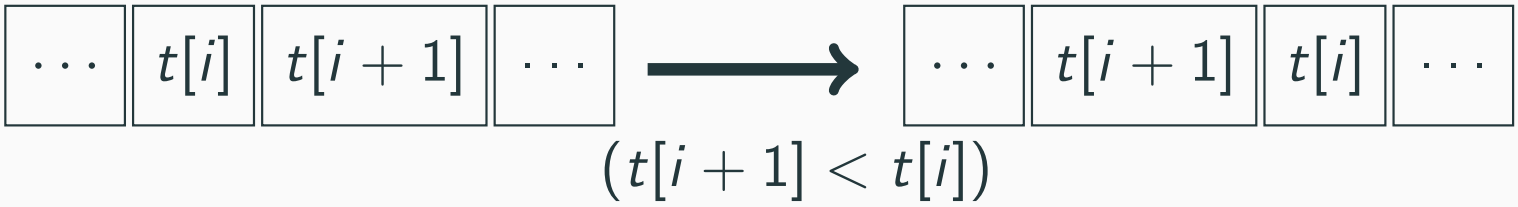
Algorithms de tri avancés

- Tri rapide (quicksort)
- Tri Shell
- Tri par tas (heapsort)

Révision

Tri à bulles

Une opération simple : permutation d'éléments adjacents

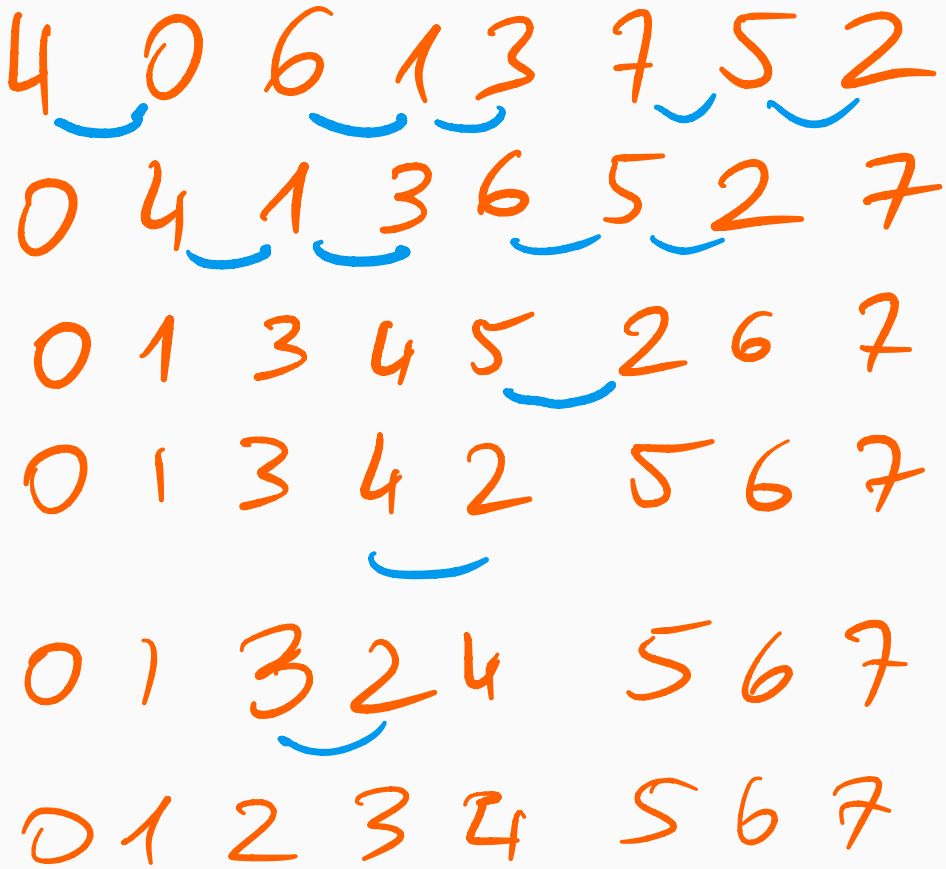


Algorithme TriBulleBase

```
1: for 1 ≤ b < n do
2:     for 0 ≤ i < n − b do
3:         if t[i + 1] < t[i] then
4:             Permute(t[i + 1], t[i])
5:         end if
6:     end for
7: end for
```

Exercice

Trier $t = [4, 0, 6, 1, 3, 7, 5, 2]$ en utilisant le tri à bulles.



Exercice

Trier $t = [4, 0, 6, 1, 3, 7, 5, 2]$ en utilisant le tri à bulles.

Tri par sélection

Opération : sélectionner le plus grand élément :

- On cherche dans le tableau le plus grand élément
- On permute cet élément avec le dernier élément du tableau

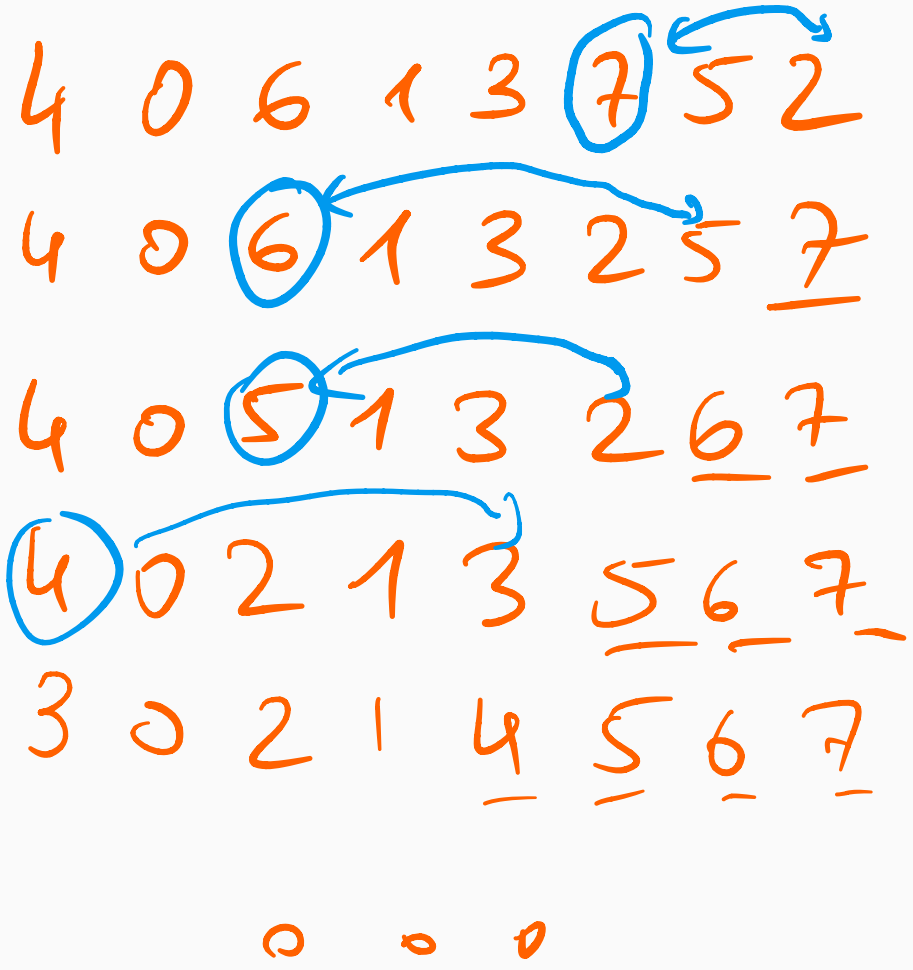
($t[i]$ maximal)

Algorithme TriSelection

```
1: while n > 1 do
2:     p ← IndiceMax(t, n)
3:     Permute(t[p], t[n − 1])
4:     n ← n − 1
5: end while
```

Exercice

Trier $t = [4, 0, 6, 1, 3, 7, 5, 2]$ en utilisant le tri par sélection.

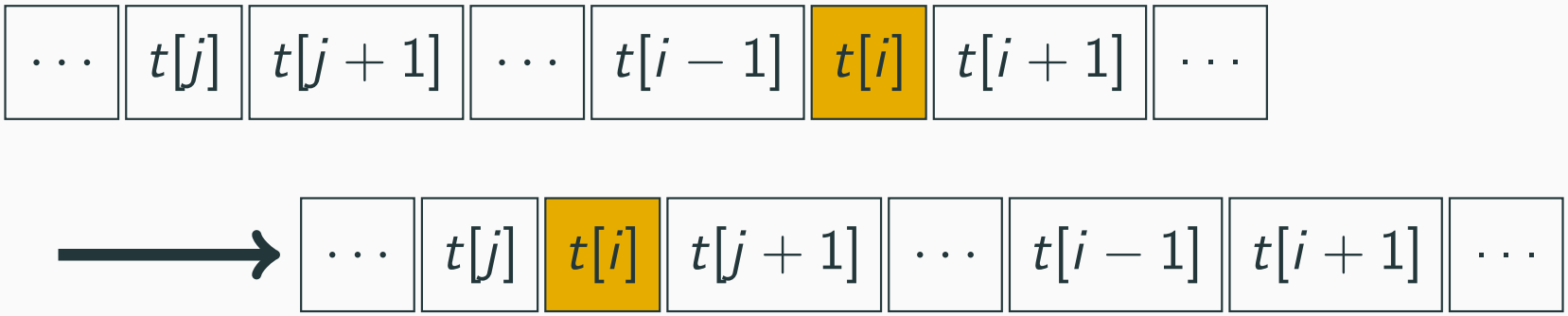


Exercice

Trier $t = [4, 0, 6, 1, 3, 7, 5, 2]$ en utilisant le tri par sélection.

Tri par insertion

Opération : insérer l'élément $t[i]$ dans $t[0: i+1]$



$(t[j] \leq t[i] < t[j+1])$

Tri par insertion

Algorithme TriInsertion

```
1: for 1 ≤ i < n do
2:     x ← t[i]
3:     p ← Recherche(t, i, x)
4:     Decalage(t, p, i − 1)
5:     t[p] ← x
6: end for
```

position pour x dans t[0..i].

Complexité pour les tableaux presque triés dépend de la recherche

- Peut être $\Theta(n \log n)$ (dichotomique) ou $\Theta(n^2)$ (linéaire)

Tri par insertion

(examen)

Algorithme TriInsertion

```
1: for 1 ≤ i < n do
2:     x ← t[i]
3:     j ← i
4:     while j > 0 ∧ t[j − 1] > x do
5:         t[j] ← t[j − 1]
6:         j ← j − 1
7:     end while
8:     t[j] ← x
9: end for
```

for i de 1 à n-1

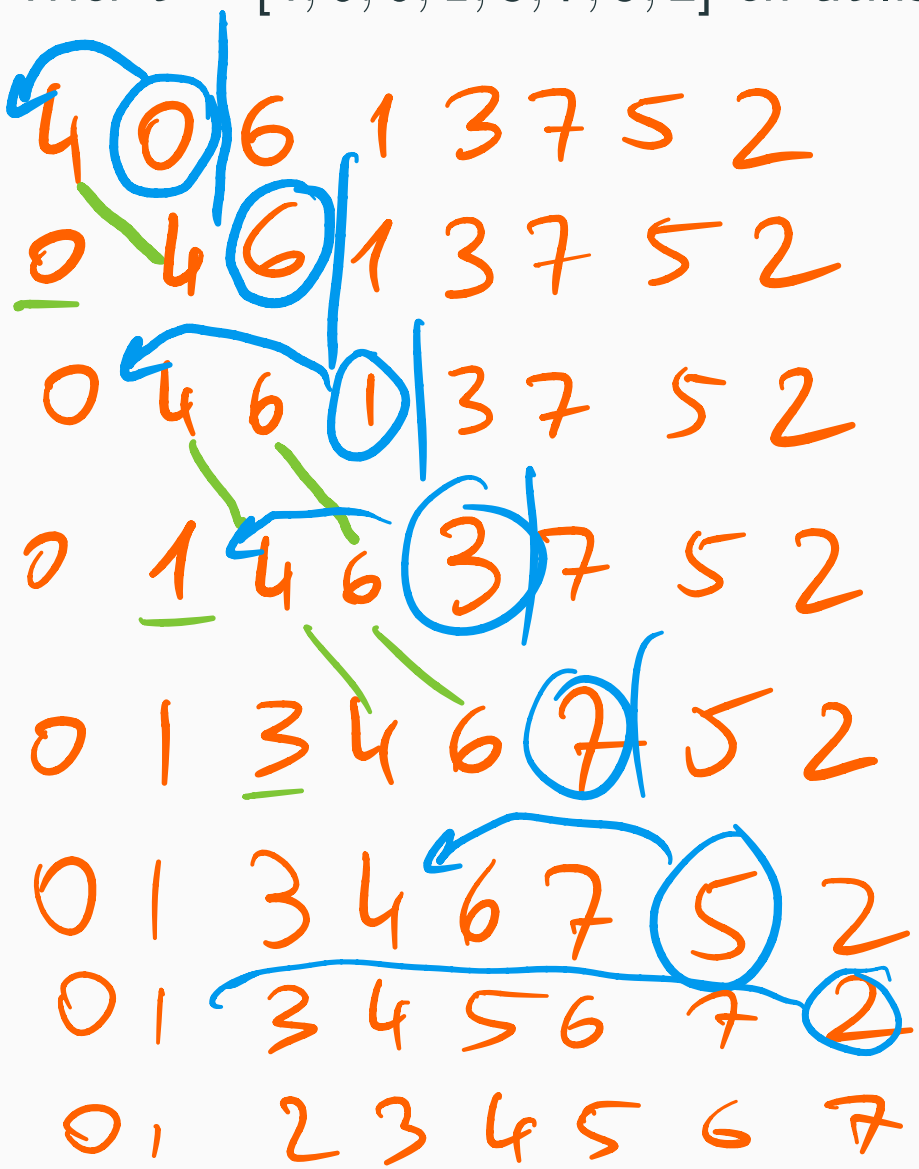
et

Complexité dans le meilleur cas : $\Theta(n)$

- Pourquoi ?

Exercice

Trier $t = [4, 0, 6, 1, 3, 7, 5, 2]$ en utilisant le tri par insertion.



Exercice

Trier $t = [4, 0, 6, 1, 3, 7, 5, 2]$ en utilisant le tri par insertion.

Complexité

	Meilleur cas	Moyen cas	Pire cas
Tri bulles*	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$
Tri sélection	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$
Tri insertion	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$

Algorithms de tri avancés

Algorithms de tri avancés

Tri rapide (quicksort)

Tri rapide

Méthode : Diviser pour régner

Idée : trier récursivement les éléments inférieurs et supérieurs

- Un élément du tableau est choisi (le pivot),
- Les éléments inférieurs au pivot sont déplacés vers sa gauche,
- Et les éléments supérieurs à sa droite,

Tri rapide

Méthode : Diviser pour régner

Idée : trier récursivement les éléments inférieurs et supérieurs

- Un élément du tableau est choisi (le pivot),
- Les éléments inférieurs au pivot sont déplacés vers sa gauche,
- Et les éléments supérieurs à sa droite,

- Ensuite, les deux parties sont triées récursivement.

Tri rapide

Méthode : Diviser pour régner

Idée : trier récursivement les éléments inférieurs et supérieurs

- Un élément du tableau est choisi (le pivot),
- Les éléments inférieurs au pivot sont déplacés vers sa gauche,
- Et les éléments supérieurs à sa droite,

- Ensuite, les deux parties sont triées récursivement.

Très populaire, car il fonctionne bien en pratique

Tri rapide

Algorithme TriRapide

Données: tableau t de type s, début d et fin f

trier $t[d..f]$ (de d à $f-1$)

1: **if** $f - d > 1$ **then**
2: $p \leftarrow \textsc{Partition}(t, d, f)$
// Sélectionne le pivot, regroupe éléments inférieurs / supérieurs
3: $\textsc{TriRapide}(t, d, p)$
4: $\textsc{TriRapide}(t, p + 1, f)$
5: **end if**

Tri rapide

Algorithme TriRapide

Données: tableau t de type s, début d et fin f

1: **if** $f - d > 1$ **then**
2: $p = \textsc{Partition}(t, d, f)$
// Sélectionne le pivot, regroupe éléments inférieurs / supérieurs
3: $\textsc{TriRapide}(t, d, p)$
4: $\textsc{TriRapide}(t, p + 1, f)$
5: **end if**

Le tri initial est appelé avec $\textsc{TriRapide}(t, 0, n)$

Partition (annexe tri rapide)

Problème : placer les petits devant et les grands derrière ;
retourner la position de la frontière (le pivot)

Partition (annexe tri rapide)

Problème : placer les petits devant et les grands derrière ;
retourner la position de la frontière (le pivot)

Comment choisir le pivot ? Plusieurs stratégies possibles.
Ici : le premier élément

Partition (annexe tri rapide)

Algorithme Partition

Données: tableau t de type s, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d$, $d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: $\textsc{Permute}(t[d], t[f - 1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: $\textsc{Permute}(t[p], t[d - 1])$
11: **return** $d - 1$

Exécution de l'algorithme de partition

0	1	2	3	4	5	6	7
4	0	6	1	3	7	5	2

$p = 0$

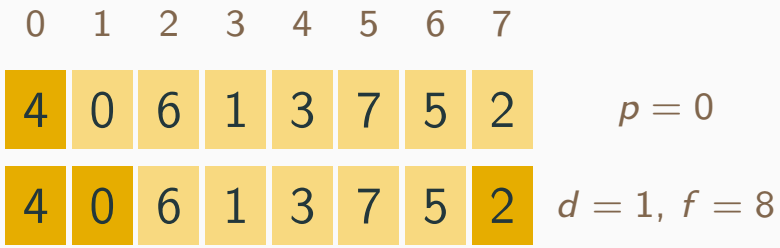
Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

```
1:  p ← d, d ← d + 1
2:  while d < f do
3:      if t[d] ≤ t[p] then
4:          d ← d + 1
5:      else
6:          PERMUTE(t[d], t[f − 1])
7:          f ← f − 1
8:      end if
9:  end while
10: PERMUTE(t[p], t[d − 1])
11: return d − 1
```

Exécution de l'algorithme de partition



Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d,\ d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: $\textsc{Permute}(t[d], t[f-1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: $\textsc{Permute}(t[p], t[d-1])$
11: **return** $d - 1$

Exécution de l'algorithme de partition

Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d,\ d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: $\textsc{Permute}(t[d], t[f-1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: $\textsc{Permute}(t[p], t[d-1])$
11: **return** $d - 1$

Exécution de l'algorithme de partition

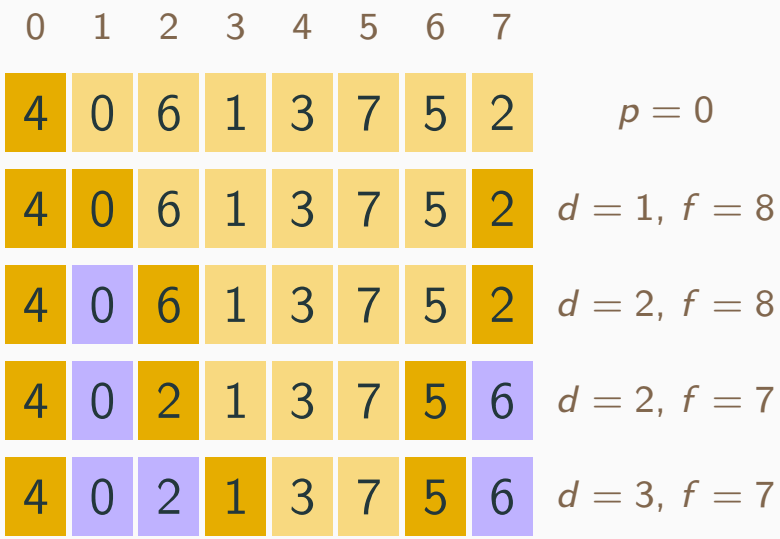
Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d$, $d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: $\textsc{Permute}(t[d], t[f-1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: $\textsc{Permute}(t[p], t[d-1])$
11: **return** $d - 1$

Exécution de l'algorithme de partition



Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d,\ d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: $\textsc{Permute}(t[d], t[f-1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: $\textsc{Permute}(t[p], t[d-1])$
11: **return** $d - 1$

Exécution de l'algorithme de partition

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$p = 0$
4	0	6	1	3	7	5	2	$d = 1,\ f = 8$
4	0	6	1	3	7	5	2	$d = 2,\ f = 8$
4	0	2	1	3	7	5	6	$d = 2,\ f = 7$
4	0	2	1	3	7	5	6	$d = 3,\ f = 7$
4	0	2	1	3	7	5	6	$d = 4,\ f = 7$

Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d$, $d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: $\textsc{Permute}(t[d], t[f - 1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: $\textsc{Permute}(t[p], t[d - 1])$
11: **return** $d - 1$

Exécution de l'algorithme de partition

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$p = 0$
4	0	6	1	3	7	5	2	$d = 1,\ f = 8$
4	0	6	1	3	7	5	2	$d = 2,\ f = 8$
4	0	2	1	3	7	5	6	$d = 2,\ f = 7$
4	0	2	1	3	7	5	6	$d = 3,\ f = 7$
4	0	2	1	3	7	5	6	$d = 4,\ f = 7$
4	0	2	1	3	7	5	6	$d = 5,\ f = 7$

Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d$, $d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: $\textsc{Permute}(t[d], t[f - 1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: $\textsc{Permute}(t[p], t[d - 1])$
11: **return** $d - 1$

Exécution de l'algorithme de partition

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$p = 0$
4	0	6	1	3	7	5	2	$d = 1,\ f = 8$
4	0	6	1	3	7	5	2	$d = 2,\ f = 8$
4	0	2	1	3	7	5	6	$d = 2,\ f = 7$
4	0	2	1	3	7	5	6	$d = 3,\ f = 7$
4	0	2	1	3	7	5	6	$d = 4,\ f = 7$
4	0	2	1	3	7	5	6	$d = 5,\ f = 7$
4	0	2	1	3	5	7	6	$d = 5,\ f = 6$

Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

```
1:  p ← d, d ← d + 1
2:  while d < f do
3:      if t[d] ≤ t[p] then
4:          d ← d + 1
5:      else
6:          PERMUTE(t[d], t[f − 1])
7:          f ← f − 1
8:      end if
9:  end while
10: PERMUTE(t[p], t[d − 1])
11: return d − 1
```

Exécution de l'algorithme de partition

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$p = 0$
4	0	6	1	3	7	5	2	$d = 1,\ f = 8$
4	0	6	1	3	7	5	2	$d = 2,\ f = 8$
4	0	2	1	3	7	5	6	$d = 2,\ f = 7$
4	0	2	1	3	7	5	6	$d = 3,\ f = 7$
4	0	2	1	3	7	5	6	$d = 4,\ f = 7$
4	0	2	1	3	7	5	6	$d = 5,\ f = 7$
4	0	2	1	3	5	7	6	$d = 5,\ f = 6$
4	0	2	1	3	5	7	6	$d = 5,\ f = 5$

Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d$, $d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: $\textsc{Permute}(t[d], t[f - 1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: $\textsc{Permute}(t[p], t[d - 1])$
11: **return** $d - 1$

Exécution de l'algorithme de partition

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$p = 0$
4	0	6	1	3	7	5	2	$d = 1,\ f = 8$
4	0	6	1	3	7	5	2	$d = 2,\ f = 8$
4	0	2	1	3	7	5	6	$d = 2,\ f = 7$
4	0	2	1	3	7	5	6	$d = 3,\ f = 7$
4	0	2	1	3	7	5	6	$d = 4,\ f = 7$
4	0	2	1	3	7	5	6	$d = 5,\ f = 7$
4	0	2	1	3	5	7	6	$d = 5,\ f = 6$
4	0	2	1	3	5	7	6	$d = 5,\ f = 5$
3	0	2	1	4	5	7	6	

Algorithme Partition

Données: tableau t, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d$, $d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: PERMUTE$(t[d], t[f-1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: PERMUTE$(t[p], t[d-1])$
11: **return** $d - 1$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$
0	1	2	3	4	5	7	6	$d = 2,\ f = 3$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$
0	1	2	3	4	5	7	6	$d = 2,\ f = 3$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$
0	1	2	3	4	5	7	6	$d = 2,\ f = 3$
0	1	2	3	4	5	7	6	$d = 4,\ f = 4$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$
0	1	2	3	4	5	7	6	$d = 2,\ f = 3$
0	1	2	3	4	5	7	6	$d = 4,\ f = 4$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$
0	1	2	3	4	5	7	6	$d = 2,\ f = 3$
0	1	2	3	4	5	7	6	$d = 4,\ f = 4$

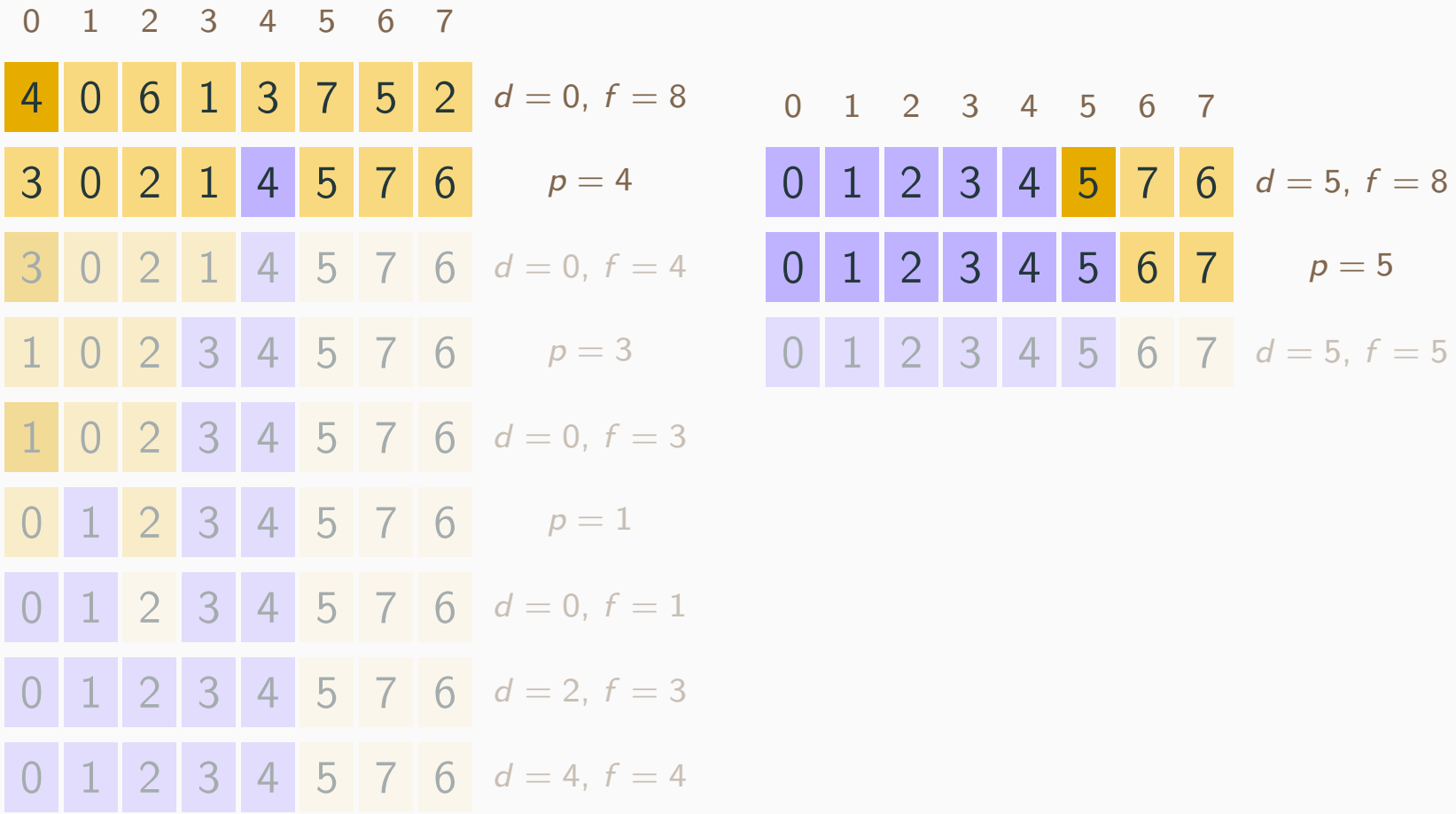
0	1	2	3	4	5	6	7	
0	1	2	3	4	5	7	6	$d = 5,\ f = 8$
0	1	2	3	4	5	6	7	$p = 5$

Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$
0	1	2	3	4	5	7	6	$d = 2,\ f = 3$
0	1	2	3	4	5	7	6	$d = 4,\ f = 4$

0	1	2	3	4	5	6	7	
0	1	2	3	4	5	7	6	$d = 5,\ f = 8$
0	1	2	3	4	5	6	7	$p = 5$
0	1	2	3	4	5	6	7	$d = 5,\ f = 5$

Exécution d'un tri rapide

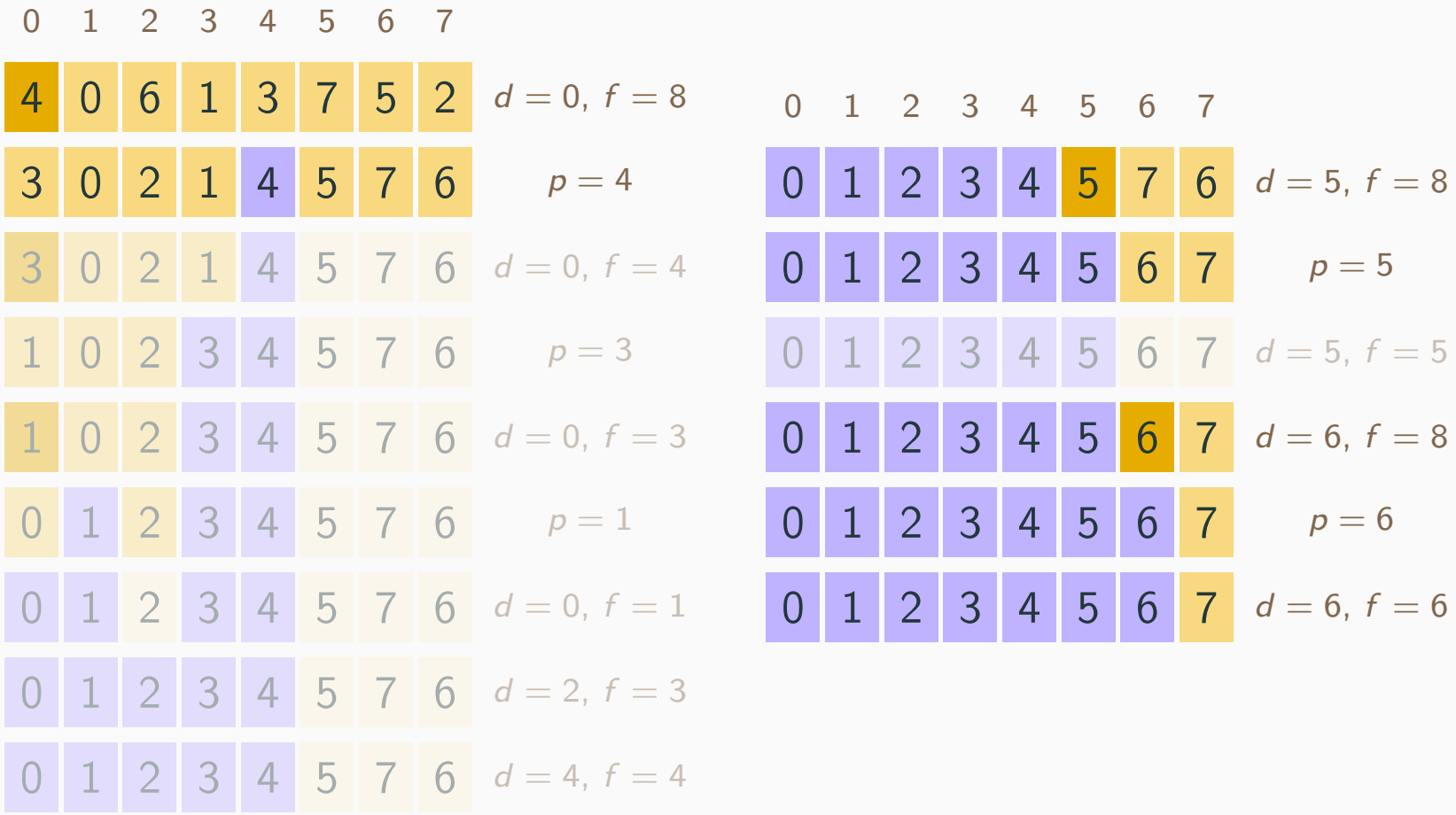


Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$
0	1	2	3	4	5	7	6	$d = 2,\ f = 3$
0	1	2	3	4	5	7	6	$d = 4,\ f = 4$

0	1	2	3	4	5	6	7	
0	1	2	3	4	5	7	6	$d = 5,\ f = 8$
0	1	2	3	4	5	6	7	$p = 5$
0	1	2	3	4	5	6	7	$d = 5,\ f = 5$
0	1	2	3	4	5	6	7	$d = 6,\ f = 8$
0	1	2	3	4	5	6	7	$p = 6$

Exécution d'un tri rapide

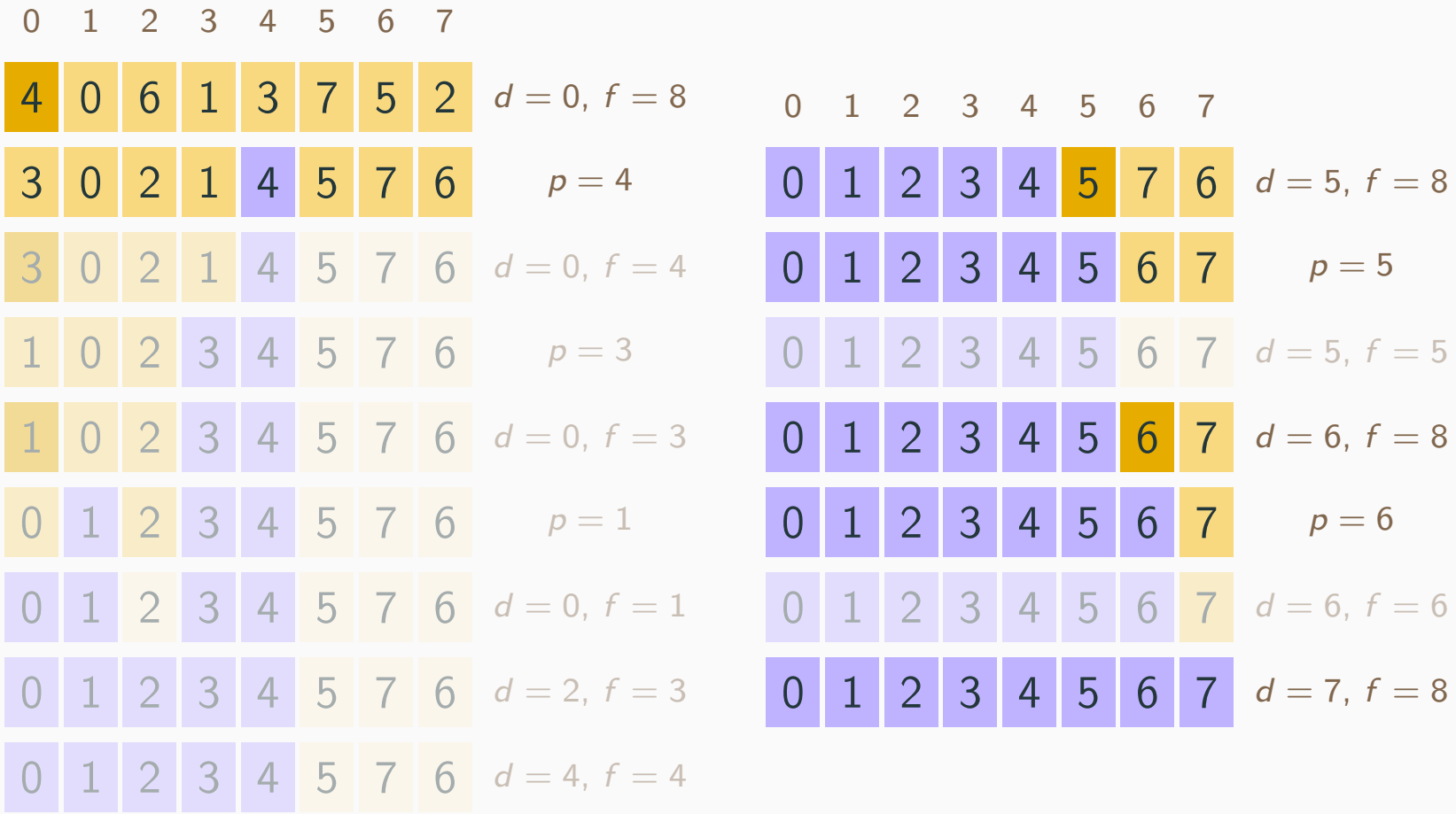


Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$
0	1	2	3	4	5	7	6	$d = 2,\ f = 3$
0	1	2	3	4	5	7	6	$d = 4,\ f = 4$

0	1	2	3	4	5	6	7	
0	1	2	3	4	5	7	6	$d = 5,\ f = 8$
0	1	2	3	4	5	6	7	$p = 5$
0	1	2	3	4	5	6	7	$d = 5,\ f = 5$
0	1	2	3	4	5	6	7	$d = 6,\ f = 8$
0	1	2	3	4	5	6	7	$p = 6$
0	1	2	3	4	5	6	7	$d = 6,\ f = 6$

Exécution d'un tri rapide

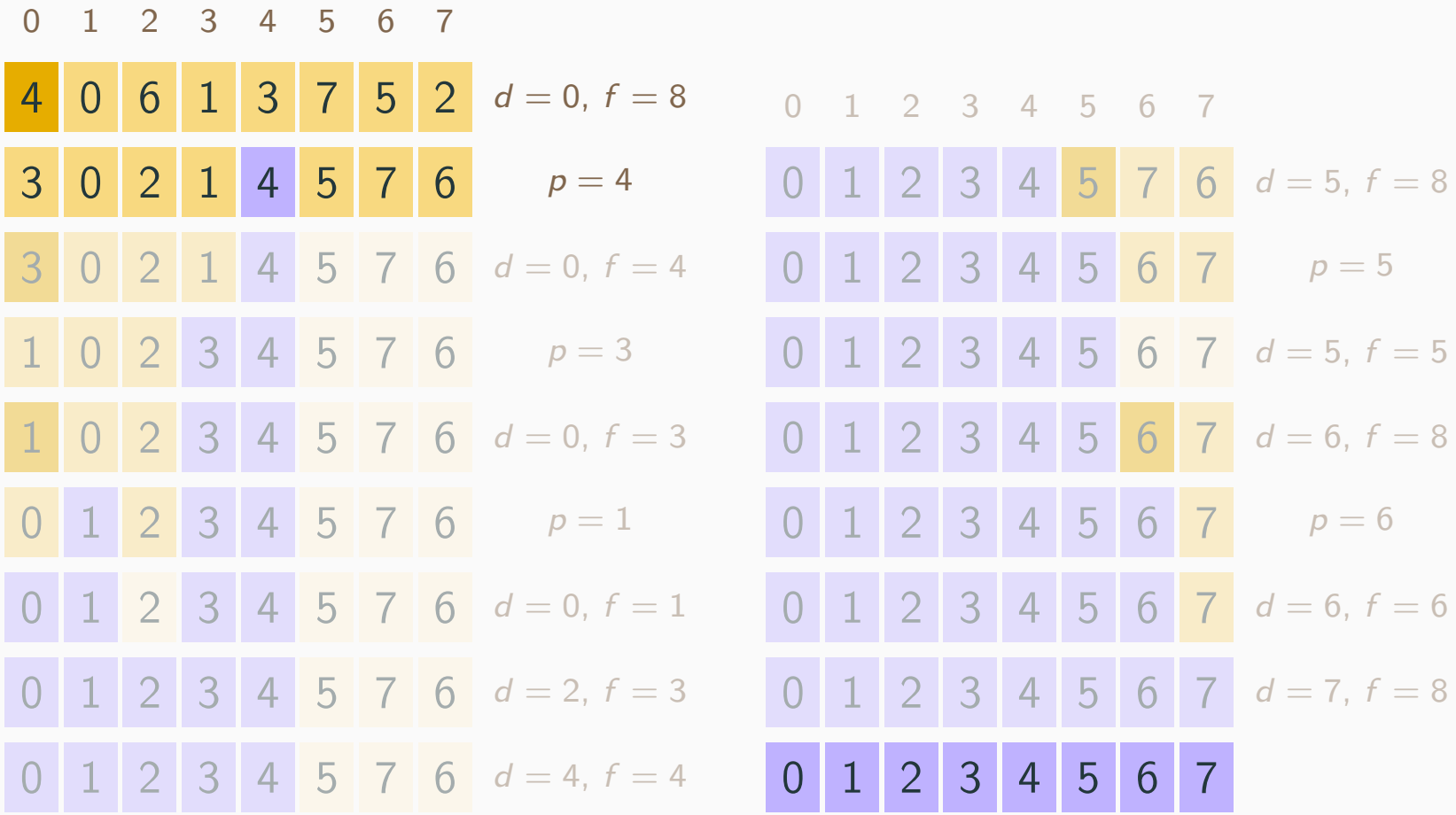


Exécution d'un tri rapide

0	1	2	3	4	5	6	7	
4	0	6	1	3	7	5	2	$d = 0,\ f = 8$
3	0	2	1	4	5	7	6	$p = 4$
3	0	2	1	4	5	7	6	$d = 0,\ f = 4$
1	0	2	3	4	5	7	6	$p = 3$
1	0	2	3	4	5	7	6	$d = 0,\ f = 3$
0	1	2	3	4	5	7	6	$p = 1$
0	1	2	3	4	5	7	6	$d = 0,\ f = 1$
0	1	2	3	4	5	7	6	$d = 2,\ f = 3$
0	1	2	3	4	5	7	6	$d = 4,\ f = 4$

0	1	2	3	4	5	6	7	
0	1	2	3	4	5	7	6	$d = 5,\ f = 8$
0	1	2	3	4	5	6	7	$p = 5$
0	1	2	3	4	5	6	7	$d = 5,\ f = 5$
0	1	2	3	4	5	6	7	$d = 6,\ f = 8$
0	1	2	3	4	5	6	7	$p = 6$
0	1	2	3	4	5	6	7	$d = 6,\ f = 6$
0	1	2	3	4	5	6	7	$d = 7,\ f = 8$

Exécution d'un tri rapide



Complexité

Algorithme TriRapide

Données: tableau t de type s, début d et fin f

1: **if** $f - d > 1$ **then**
2: $p = \text{PARTITION}(t, d, f)$ $\Theta(n)$
// *Sélectionne le pivot, regroupe éléments inférieurs / supérieurs*
3: $\text{TRIRAPIDE}(t, d, p)$
4: $\text{TRIRAPIDE}(t, p + 1, f)$
5: **end if**

Complexité ? pire cas $\Theta(n^2)$

Cas moyen → imaginer 2 sous-problèmes de taille $n/2$
→ $\Theta(n \log n)$

Algorithme Partition

Données: tableau t de type s, début d et fin f

Résultat: position du pivot p

1: $p \leftarrow d$, $d \leftarrow d + 1$
2: **while** $d < f$ **do**
3: **if** $t[d] \leq t[p]$ **then**
4: $d \leftarrow d + 1$
5: **else**
6: $\textsc{Permute}(t[d], t[f - 1])$
7: $f \leftarrow f - 1$
8: **end if**
9: **end while**
10: $\textsc{Permute}(t[p], t[d - 1])$
11: **return** $d - 1$

Complexité

Dépend du choix du pivot, mais en général :

- Pire cas : $\Theta(n^2)$
- Cas moyen : $\Theta(n \log n)$
- Meilleur cas : $\Theta(n \log n)$

$$\log_a b = \frac{\log_2 b}{\log_2 a}$$

Pire cas peut être en $\Theta(n \log n)$:

- Pivot : médiane ($\Theta(n)$)
- Pas suffisamment efficace dans la pratique

Le tri rapide est en place, mais pas stable.

Algorithms de tri avancés

Tri Shell

Tri Shell

Optimisation du tri par insertion

- Permet l'échange d'éléments très éloignés les uns des autres

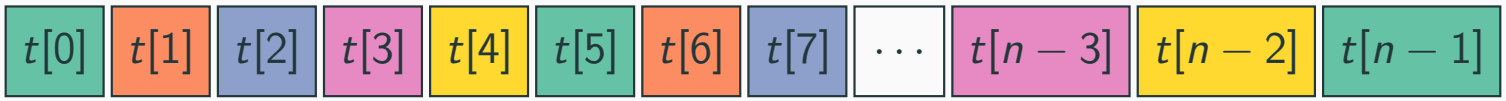
Tri Shell

Optimisation du tri par insertion

- Permet l'échange d'éléments très éloignés les uns des autres

Idéé : trier d'abord les séquences de chaque h-ème élément

- Trier chaque couleur (par insertion), $h = 5$



- Permet aux éléments de se déplacer sur de longues distances
- Laisse moins de travail au tri final

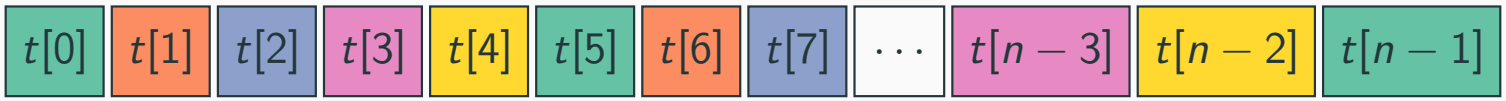
Tri Shell

Optimisation du tri par insertion

- Permet l'échange d'éléments très éloignés les uns des autres

Idéé : trier d'abord les séquences de chaque h-ème élément

- Trier chaque couleur (par insertion), $h = 5$



- Permet aux éléments de se déplacer sur de longues distances
- Laisse moins de travail au tri final

Effectue plusieurs tris sur une séquence décroissante de h

- Termine toujours par un tri $h = 1$ (tri par insertion)

Séquences de valeurs de h ?

Comment choisir les valeurs de h ?

- Shell : $\left\lfloor \frac{n}{2^i} \right\rfloor \rightarrow \left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{4} \right\rfloor, \ldots, 1$
  - $n = 50 \rightarrow 25, 12, 6, 3, 1$
- Hibbard : $2^i - 1 \rightarrow 2^{\lfloor \log n \rfloor} - 1, \ldots, 7, 3, 1$
  - $n = 50 \rightarrow 31, 15, 7, 3, 1$
- Sedgewick : $4^k + 3 \cdot 2^{k-1} + 1 \rightarrow \ldots, 281, 77, 23, 8, 1$
  - $n = 50 \rightarrow 23, 8, 1$

Séquences de valeurs de h ?

Comment choisir les valeurs de h ?

- Shell : $\left\lfloor \frac{n}{2^i} \right\rfloor \rightarrow \left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{4} \right\rfloor, \ldots, 1$
  - $n = 50 \rightarrow 25, 12, 6, 3, 1$
- Hibbard : $2^i - 1 \rightarrow 2^{\lfloor \log n \rfloor} - 1, \ldots, 7, 3, 1$
  - $n = 50 \rightarrow 31, 15, 7, 3, 1$
- Sedgewick : $4^k + 3 \cdot 2^{k-1} + 1 \rightarrow \ldots, 281, 77, 23, 8, 1$
  - $n = 50 \rightarrow 23, 8, 1$
- Tokuda : $\left\lceil \frac{4}{5} \left(\left(\frac{9}{4} \right)^i - 1 \right) \right\rceil \rightarrow \ldots, 103, 46, 20, 9, 4, 1$
  - $n = 50 \rightarrow 46, 20, 9, 4, 1$

Tri Shell

Algorithme TriShell

Données: tableau t de type s, taille du tableau n

```
1: h ← int(n/2)
2: while h > 0 do
3:     for 0 ≤ i < h do
4:         TriSousTableau(t, n, i, h)
5:     end for
6:     h ← int(h/2)
7: end while
```

↳ nombre de couleurs
↳ indice de couleur

↳ trie les éléments aux positions
i, i+h, i+2h, ...

Tri Shell (TriSousTableau)

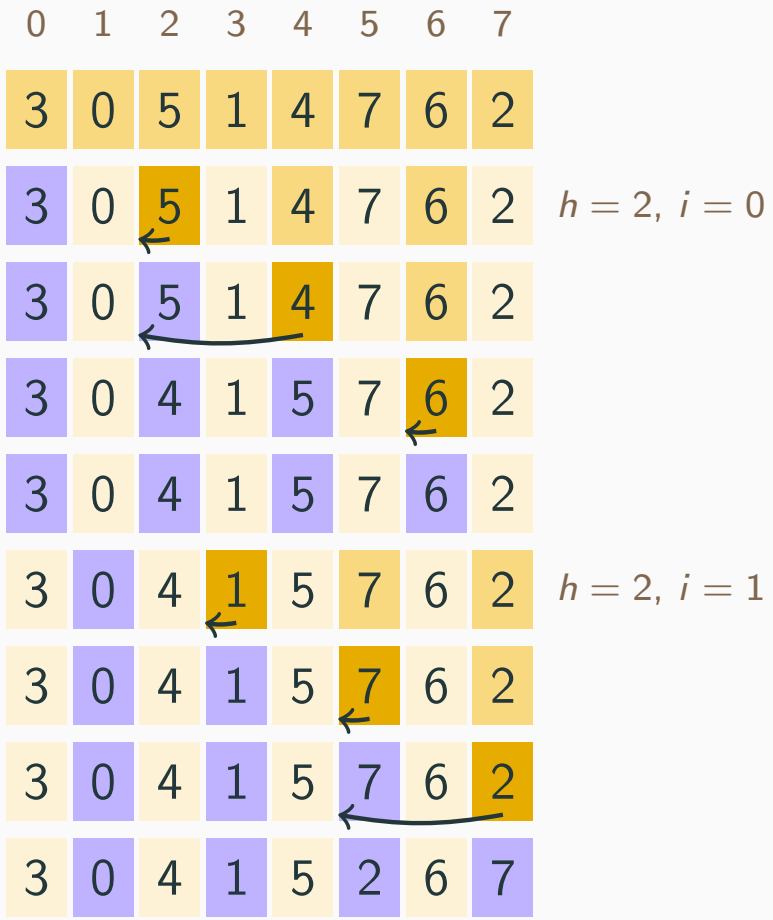
Algorithme TriSousTableau

Données: tableau t de type s, entiers n, i, h

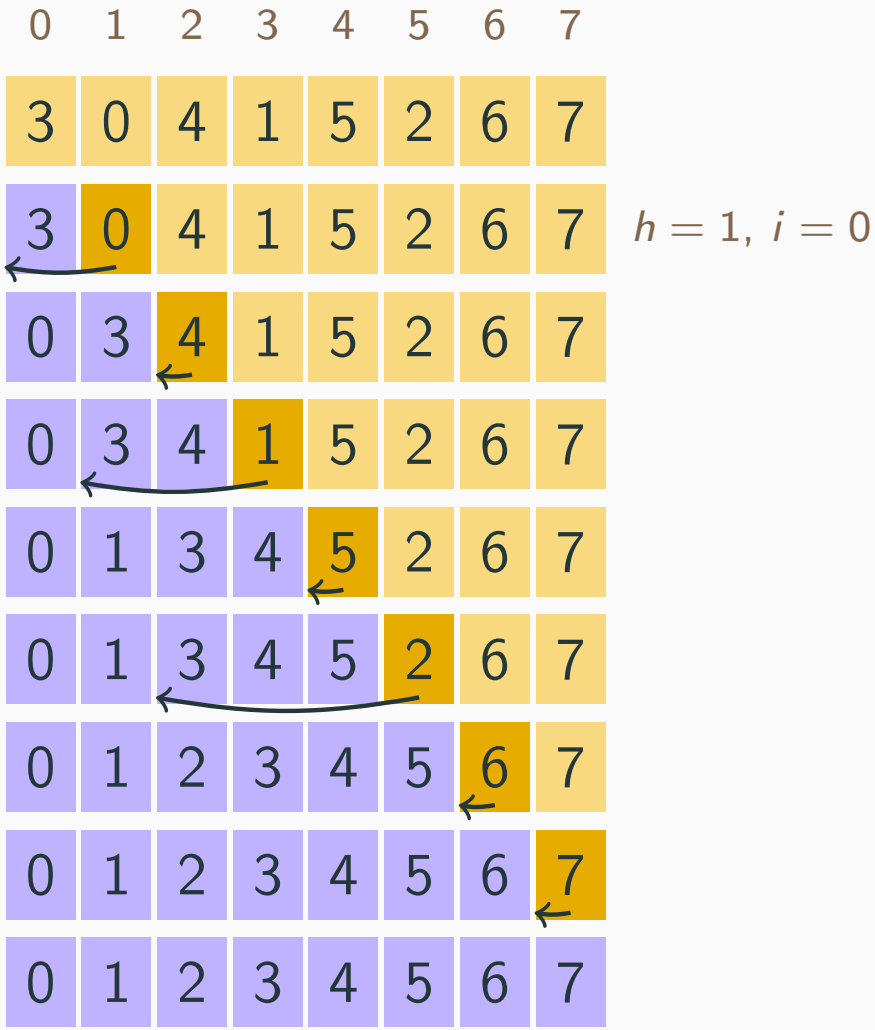
trie les éléments i, $i+h$, $i+2h$, ...

1: $i \leftarrow i + h$
2: **while** $i < n$ **do**
3: $\quad x \leftarrow t[i]$
4: $\quad j \leftarrow i$
5: $\quad$ **while** $j - h \geq 0 \wedge t[j - h] > x$ **do**
6: $\quad\quad t[j] \leftarrow t[j - h]$
7: $\quad\quad j \leftarrow j - h$
8: $\quad$ **end while**
9: $\quad t[j] \leftarrow x$; $i \leftarrow i + h$
10: **end while**

Exécution d'un tri Shell



Exécution d'un tri Shell



Complexité

Le choix de la séquence d'écart à utiliser est difficile et a un impact sur la complexité de l'algorithme.

Algorithme	Séquence	Exemples	Complexité
Shell	$\left\lfloor \frac{n}{2^i} \right\rfloor$	$25, 12, 6, 3, 1$	$\Theta(n^2)$
Hibbard	$2^i - 1$	$15, 7, 3, 1$	$\Theta(n^{3/2})$
Sedgewick	$4^k + 3 \cdot 2^{k-1} + 1$	$281, 77, 23, 8, 1$	$\Theta(n^{4/3})$
Pratt	$2^p 3^q$	$12, 9, 8, 6, 4, 3, 2, 1$	$\Theta(n \log^2 n)$
Tokuda	$\left\lceil \frac{4}{5}\left(\left(\frac{9}{4}\right)^i - 1\right)\right\rceil$	$103, 46, 20, 9, 4, 1$	?

Le tri Shell est en place, mais pas stable.

Exercice

Trier le tableau $t = [4, 0, 6, 1, 3, 7, 5, 2]$

pour différentes séquences d'écart :

- $4, 2, 1$ (Shell)
- $7, 3, 1$ (Hibbard)

Calculer et comparer le nombre de d'opérations (comparaisons) nécessaires pour exécuter l'algorithme pour ces deux alternatives.

Exercice

Trier $t = [4, 0, 6, 1, 3, 7, 5, 2]$ avec la séquence $h \in \{7, 3, 1\}$.

Algorithms de tri avancés

Tri par tas (heapsort)

Tri par tas

Idée :

- Donner au tableau une structure spécifique
- Ce qui permet de extraire l'élément maximal rapidement
- Un peu comme une amélioration du tri par sélection
  - Mais beaucoup plus rapide

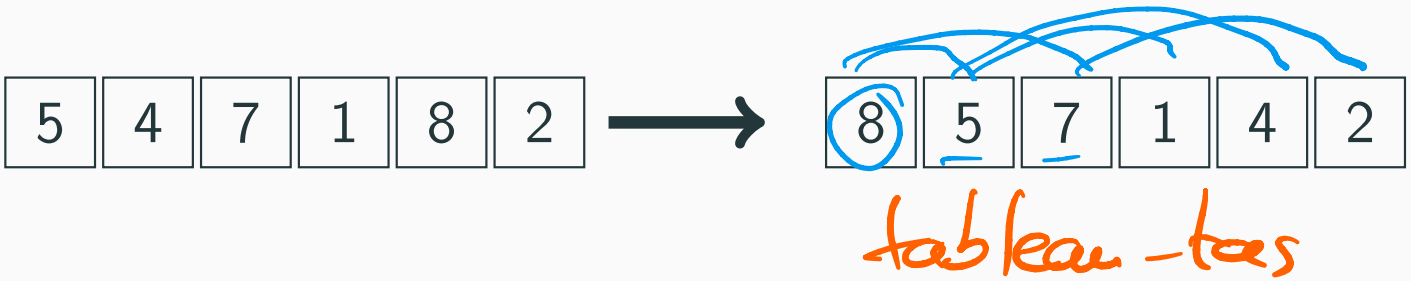


Tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

Tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

- Chaque élément a deux fils (ou moins)
    - Ce sont les deux éléments suivants disponibles,

Tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

- Chaque élément a deux fils (ou moins)
  - Ce sont les deux éléments suivants disponibles,
  - Les fils de $t[0]$ sont $t[1]$ et $t[2]$,

Tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

- Chaque élément a deux fils (ou moins)
  - Ce sont les deux éléments suivants disponibles,
  - Les fils de $t[0]$ sont $t[1]$ et $t[2]$,
  - Les fils de $t[1]$ sont $t[3]$ et $t[4]$,

Tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

- Chaque élément a deux fils (ou moins)
  - Ce sont les deux éléments suivants disponibles,
  - Les fils de $t[0]$ sont $t[1]$ et $t[2]$,
  - Les fils de $t[1]$ sont $t[3]$ et $t[4]$,
  - Les fils de $t[2]$ sont $t[5]$ et $t[6]$,

Tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

- Chaque élément a deux fils (ou moins)
  - Ce sont les deux éléments suivants disponibles,
  - Les fils de $t[0]$ sont $t[1]$ et $t[2]$,
  - Les fils de $t[1]$ sont $t[3]$ et $t[4]$,
  - Les fils de $t[2]$ sont $t[5]$ et $t[6]$,
  - Les fils de $t[3]$ sont $t[7]$ et $t[8]$,

Tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

- Chaque élément a deux fils (ou moins)
  - Ce sont les deux éléments suivants disponibles,
  - Les fils de $t[0]$ sont $t[1]$ et $t[2]$,
  - Les fils de $t[1]$ sont $t[3]$ et $t[4]$,
  - Les fils de $t[2]$ sont $t[5]$ et $t[6]$,
  - Les fils de $t[3]$ sont $t[7]$ et $t[8]$,
  - Les fils de $t[i]$ sont $t[2i+1]$ et $t[2i+2]$.

Tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

- Chaque élément a deux fils (ou moins)
  - Ce sont les deux éléments suivants disponibles,
  - Les fils de $t[0]$ sont $t[1]$ et $t[2]$,
  - Les fils de $t[1]$ sont $t[3]$ et $t[4]$,
  - Les fils de $t[2]$ sont $t[5]$ et $t[6]$,
  - Les fils de $t[3]$ sont $t[7]$ et $t[8]$,
  - Les fils de $t[i]$ sont $t[2i+1]$ et $t[2i+2]$.
- Chaque $t[i]$ doit être supérieur ou égal à ses fils

Caractéristiques d'un tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

- Chaque élément a deux fils (ou moins)
  - Ce sont les deux éléments suivants disponibles,
  - Les fils de $t[i]$ sont $t[2i+1]$ et $t[2i+2]$.
- Chaque $t[i]$ doit être supérieur ou égal à ses fils

Utilisation :

- Le maximum est toujours $t[0]$!
- On peut supprimer le dernier élément en temps $\Theta(1)$,
- On peut modifier l'élément $t[0]$ et rétablir le tas en $\Theta(\log n)$,

Caractéristiques d'un tableau-tas

Tableau-tas : tableau avec des caractéristiques spécifiques

- Chaque élément a deux fils (ou moins)
  - Ce sont les deux éléments suivants disponibles,
  - Les fils de $t[i]$ sont $t[2i+1]$ et $t[2i+2]$.
- Chaque $t[i]$ doit être supérieur ou égal à ses fils

Utilisation :

- Le maximum est toujours $t[0]$!
- On peut supprimer le dernier élément en temps $\Theta(1)$,
- On peut modifier l'élément $t[0]$ et rétablir le tas en $\Theta(\log n)$,
- On peut construire un tas à partir d'un tableau en $\Theta(n)$.

Tri par tas

Algorithme TriTas

Données: tableau t de type s, taille du tableau n

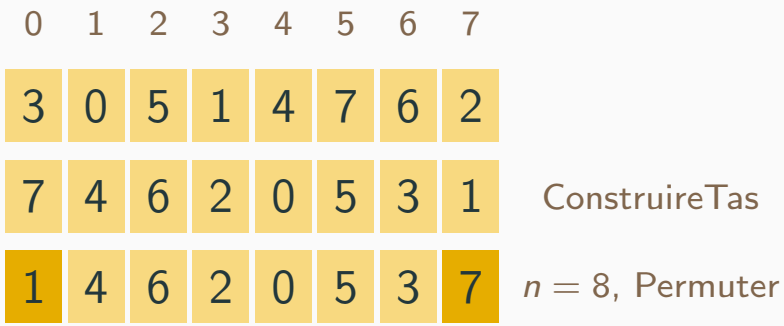
```
1: ConstruireTas(t, n)
2: while n > 1 do
3:     Permuter(t[0], t[n − 1])
4:     n ← n − 1
5:     RetablirTas(t, n, 0)
6: end while
```

Exécution d'un tri par tas

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas

Exécution d'un tri par tas



Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir
0	2	3	1	4	5	6	7	$n = 5$, Permuter

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir
0	2	3	1	4	5	6	7	$n = 5$, Permuter
3	2	0	1	4	5	6	7	$n = 4$, Rétablir

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir
0	2	3	1	4	5	6	7	$n = 5$, Permuter
3	2	0	1	4	5	6	7	$n = 4$, Rétablir

0	1	2	3	4	5	6	7	
3	2	0	1	4	5	6	7	$n = 4$

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir
0	2	3	1	4	5	6	7	$n = 5$, Permuter
3	2	0	1	4	5	6	7	$n = 4$, Rétablir

0	1	2	3	4	5	6	7	
3	2	0	1	4	5	6	7	$n = 4$
1	2	0	3	4	5	6	7	$n = 4$, Permuter

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir
0	2	3	1	4	5	6	7	$n = 5$, Permuter
3	2	0	1	4	5	6	7	$n = 4$, Rétablir

0	1	2	3	4	5	6	7	
3	2	0	1	4	5	6	7	$n = 4$
1	2	0	3	4	5	6	7	$n = 4$, Permuter
2	1	0	3	4	5	6	7	$n = 3$, Rétablir

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir
0	2	3	1	4	5	6	7	$n = 5$, Permuter
3	2	0	1	4	5	6	7	$n = 4$, Rétablir

0	1	2	3	4	5	6	7	
3	2	0	1	4	5	6	7	$n = 4$
1	2	0	3	4	5	6	7	$n = 4$, Permuter
2	1	0	3	4	5	6	7	$n = 3$, Rétablir
0	1	2	3	4	5	6	7	$n = 3$, Permuter

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir
0	2	3	1	4	5	6	7	$n = 5$, Permuter
3	2	0	1	4	5	6	7	$n = 4$, Rétablir

0	1	2	3	4	5	6	7	
3	2	0	1	4	5	6	7	$n = 4$
1	2	0	3	4	5	6	7	$n = 4$, Permuter
2	1	0	3	4	5	6	7	$n = 3$, Rétablir
0	1	2	3	4	5	6	7	$n = 3$, Permuter
1	0	2	3	4	5	6	7	$n = 2$, Rétablir

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir
0	2	3	1	4	5	6	7	$n = 5$, Permuter
3	2	0	1	4	5	6	7	$n = 4$, Rétablir

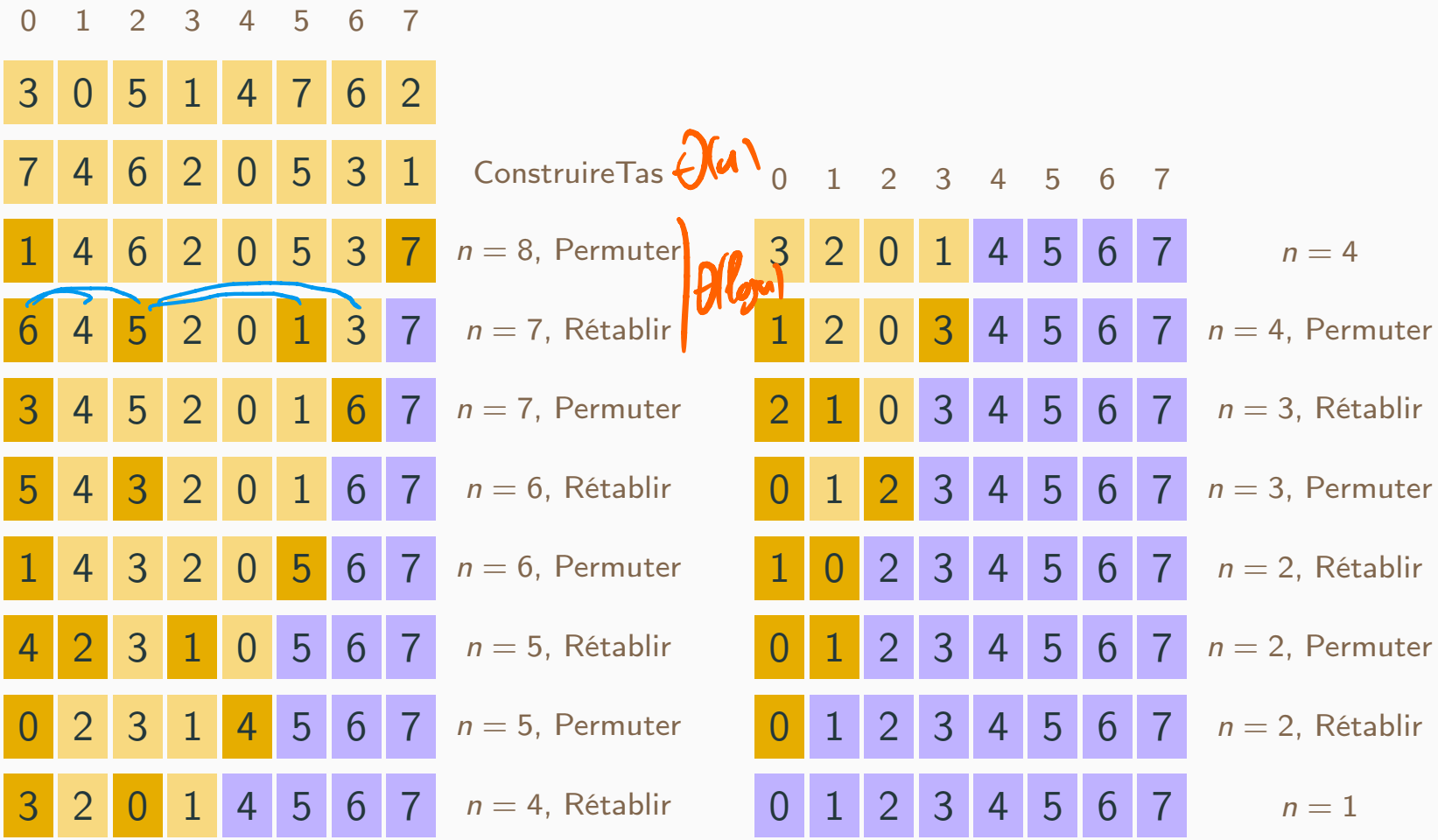
0	1	2	3	4	5	6	7	
3	2	0	1	4	5	6	7	$n = 4$
1	2	0	3	4	5	6	7	$n = 4$, Permuter
2	1	0	3	4	5	6	7	$n = 3$, Rétablir
0	1	2	3	4	5	6	7	$n = 3$, Permuter
1	0	2	3	4	5	6	7	$n = 2$, Rétablir
0	1	2	3	4	5	6	7	$n = 2$, Permuter

Exécution d'un tri par tas

0	1	2	3	4	5	6	7	
3	0	5	1	4	7	6	2	
7	4	6	2	0	5	3	1	ConstruireTas
1	4	6	2	0	5	3	7	$n = 8$, Permuter
6	4	5	2	0	1	3	7	$n = 7$, Rétablir
3	4	5	2	0	1	6	7	$n = 7$, Permuter
5	4	3	2	0	1	6	7	$n = 6$, Rétablir
1	4	3	2	0	5	6	7	$n = 6$, Permuter
4	2	3	1	0	5	6	7	$n = 5$, Rétablir
0	2	3	1	4	5	6	7	$n = 5$, Permuter
3	2	0	1	4	5	6	7	$n = 4$, Rétablir

0	1	2	3	4	5	6	7	
3	2	0	1	4	5	6	7	$n = 4$
1	2	0	3	4	5	6	7	$n = 4$, Permuter
2	1	0	3	4	5	6	7	$n = 3$, Rétablir
0	1	2	3	4	5	6	7	$n = 3$, Permuter
1	0	2	3	4	5	6	7	$n = 2$, Rétablir
0	1	2	3	4	5	6	7	$n = 2$, Permuter
0	1	2	3	4	5	6	7	$n = 2$, Rétablir

Exécution d'un tri par tas



Complexité : $\Theta(n \log n)$

n itérations
complex./itération : $\Theta(\log n)$

Opérations sur un tableau-tas

Comment mettre en œuvre les opérations nécessaires ?

Opérations sur un tableau-tas

Comment mettre en œuvre les opérations nécessaires ?

ConstruireTas :

- En utilisant RetablirTas
- Appeler RetablirTas pour chaque position de $n - 1$ à 0

Opérations sur un tableau-tas

Comment mettre en œuvre les opérations nécessaires ?

ConstruireTas :

- En utilisant RetablirTas
- Appeler RetablirTas pour chaque position de $n - 1$ à 0

RetablirTas sur une position i :

- Si $t[i]$ est supérieur ou égal à ses fils, terminer. $2i+1$, $2i+2$
- Permuter $t[i]$ avec le plus grand des fils,
- Répéter à la position du fils permuté.

ConstruireTas (tri par tas)

Algorithme ConstruireTas

Données: tableau t de type s, taille du tableau n

1: **for** $i = n - 1, \ldots, 0$ **do**
2: $\quad$ RetablirTas(t, n, i)
3: **end for**

RetablirTas (tri par tas)

Algorithme RetablirTas

Données: tableau t de type s, taille n, position k

```
1:  while 2k + 1 < n do
2:      j ← 2k + 1
3:      if j < n − 1 ∧ t[j] < t[j + 1] then
4:          j ← j + 1                          // fils droite plus grand
5:      end if
6:      if t[k] < t[j] then
7:          PERMUTER(t[k], t[j])
8:          k ← j
9:      else
10:         k ← n      break                   // Pour sortir de la boucle
11:     end if
12: end while
```

Complexité

Algorithme TriTas

Données: tableau t de type s, taille du tableau n

```
1: ConstruireTas(t, n)
2: while n > 1 do
3:     Permuter(t[0], t[n − 1])
4:     n ← n − 1
5:     RetablirTas(t, n, 0)
6: end while
```

Complexité ?

Complexité

- Pire cas : $\Theta(n \log n)$
- Cas moyen : $\Theta(n \log n)$
- Meilleur cas : $\Theta(n \log n)$

Le tri par tas est en place, mais pas stable.

Orange Pomme Poire

Comparaison de tris

Source : Wikipedia

Name	Best	Average	Worst	Memory	Stable	Method
In-place merge sort	—	—	$n \log^2 n$	1	Yes	Merging
Heapsort	$n \log n$	$n \log n$	$n \log n$	1	No	Selection
Introsort	$n \log n$	$n \log n$	$n \log n$	$\log n$	No	Partitioning & Selection
Merge sort	$n \log n$	$n \log n$	$n \log n$	n	Yes	Merging
Smoothsort	n	$n \log n$	$n \log n$	1	No	Selection
Timsort	n	$n \log n$	$n \log n$	n	Yes	Insertion & Merging
Patience sorting	n	$n \log n$	$n \log n$	n	No	Insertion & Selection
Quicksort	$n \log n$	$n \log n$	n^2	$\log n$	No	Partitioning
Library sort	$n \log n$	$n \log n$	n^2	n	No	Insertion
Shellsort	$n \log n$	$n^{4/3}$	$n^{3/2}$	1	No	Insertion
Comb sort	$n \log n$	n^2	n^2	1	No	Exchanging
Insertion sort	n	n^2	n^2	1	Yes	Insertion
Bubble sort	n	n^2	n^2	1	Yes	Exchanging
Cocktail shaker sort	n	n^2	n^2	1	Yes	Exchanging
Gnome sort	n	n^2	n^2	1	Yes	Exchanging
Selection sort	n^2	n^2	n^2	1	No	Selection
Exchange sort	n^2	n^2	n^2	1	No	Exchanging

tri tas

tri rapide

tri Shell

tri insertion

tri bulles

tri selection