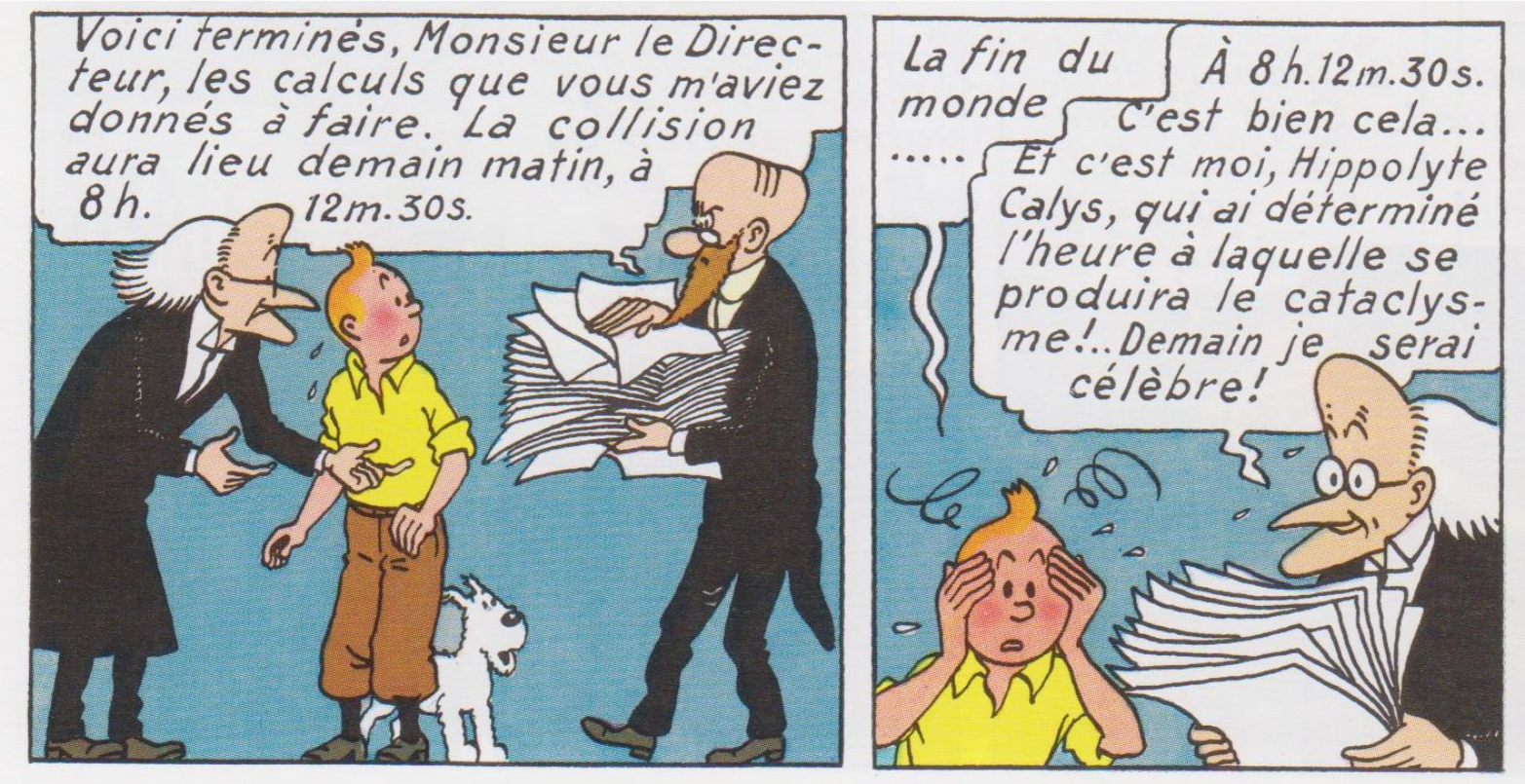
Des algorithmes pour l'astronomie?

Roger Mansuy

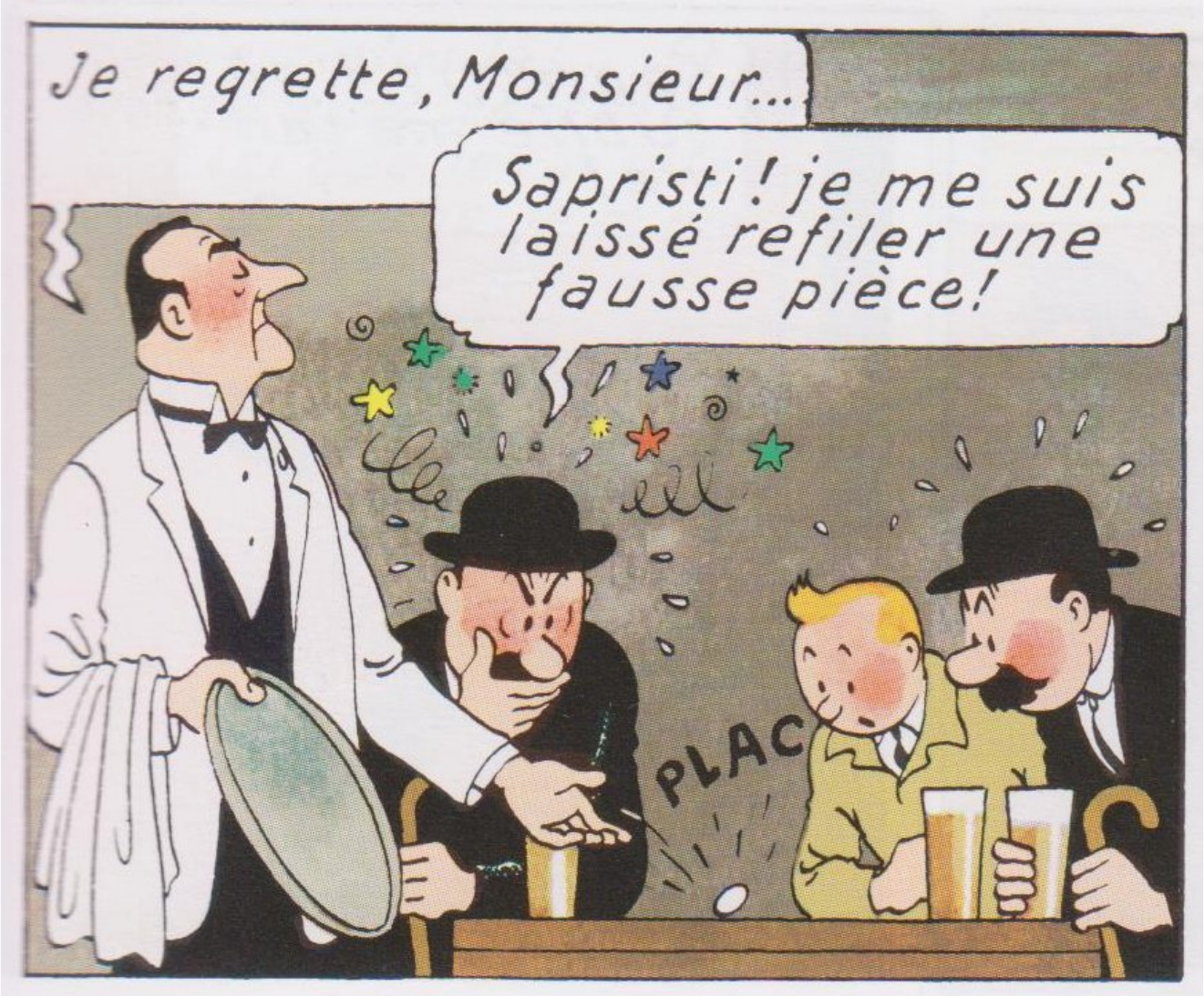
FAsF16

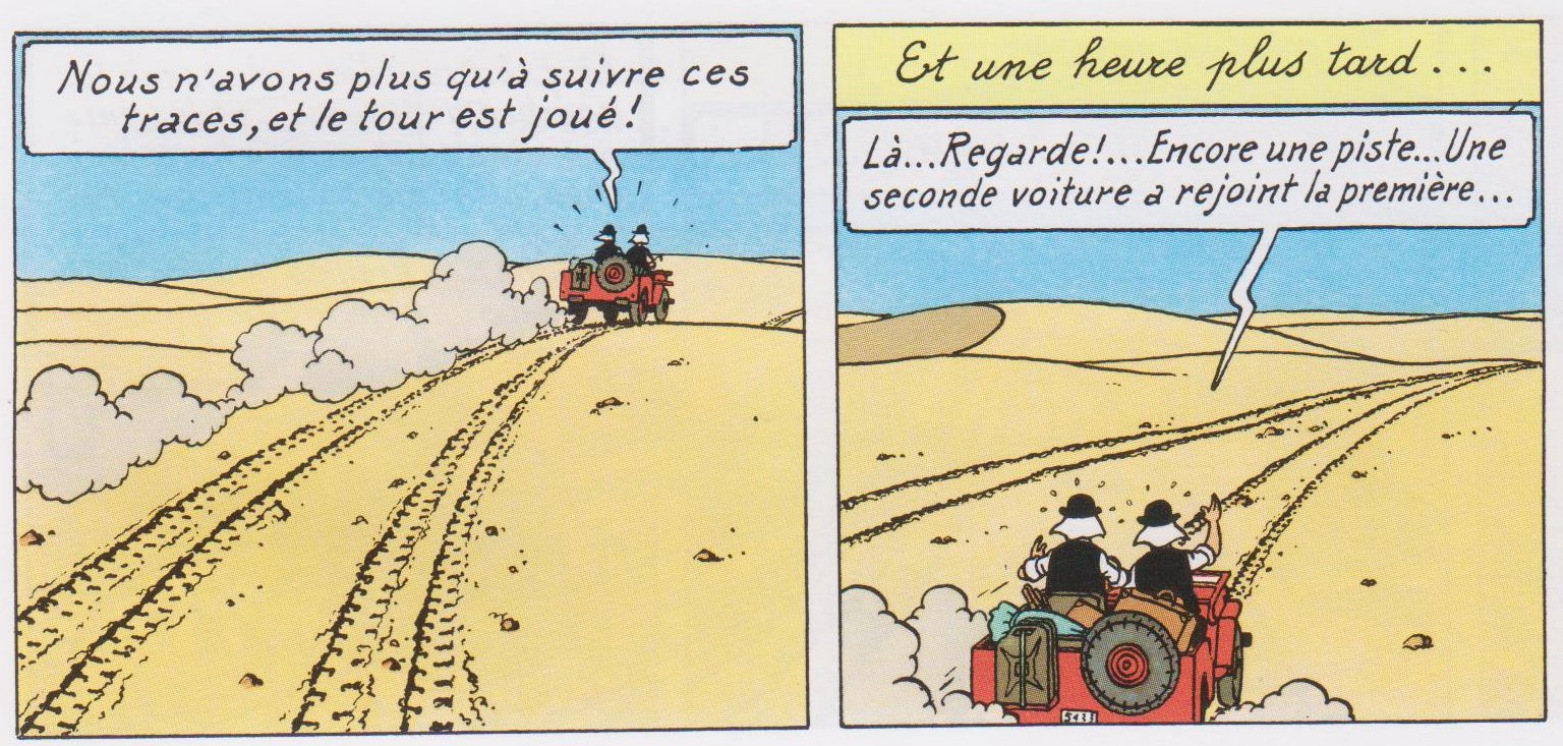
2016


Oui noble Fils du Soleil... Je désire... nous désirons mourir dans... oui, dans dix-huit jours, à 11 h... C'est l'anniversaire de naissance de mon ami le capitaine et je...
?


Voici terminés, Monsieur le Directeur, les calculs que vous m'aviez donnés à faire. La collision aura lieu demain matin, à 8 h. 12 m. 30 s.
La fin du monde .....
À 8 h. 12 m. 30 s. C'est bien cela... Et c'est moi, Hippolyte Calys, qui ai déterminé l'heure à laquelle se produira le cataclysme!.. Demain je serai célèbre!


Observatoire à Station Contrôle... La Fusée est à 800 km. de la Terre. Le moteur auxiliaire vient de s'arrêter, relayé automatiquement par le moteur atomique.
Entendu... Nous allons essayer d'entrer en contact avec la Fusée...


Je regrette, Monsieur...
Sapristi ! je me suis laissé refiler une fausse pièce !
PLAC


Nous n'avons plus qu'à suivre ces traces, et le tour est joué !
Et une heure plus tard . . .
Là...Regarde !...Encore une piste...Une seconde voiture a rejoint la première...


Puis, pendant la traversée, j'ai fini par trouver. Il suffit de prendre les deux premières lettres de chaque mot, et cela donne: "RRCQ 15.30 Envoyons marchandise. A l'écoute semaine prochaine."
Le mot "Yokohama" m'incite à croire que l'auteur de ce message est un Japonais et qu'il... Attention! un radio sur la même longueur d'onde...
cercle
solaire
irlandais
direct
hemisphère
uranus
est long
tube
blanc
eugene


Allons ! Je vais me rendre dans la pièce voisine et, l'un après l'autre, vous viendrez me dire le mot de passe de notre précé- dente réunion.
Au premier.

Question

Qu'est-ce qu'un algorithme ?

Les algorithmes vus dans la presse

- Recherche Google : PageRank
- Publicité Google : AdWords
- Fil d'actualité Facebook
- Recommandation commerciale
- Surveillance automatisée (NSA)
- Compression d'images/de vidéos
- Voitures sans conducteur
- Trading haute fréquence

Les algorithmes vus par un scientifique (Barry A. Cipra)

- 1946 : The Metropolis Algorithm for Monte Carlo
- 1947 : Simplex Method for Linear Programming
- 1950 : Krylov Subspace Iteration Method
- 1951 : The Decompositional Approach to Matrix Computations
- 1957 : The Fortran Optimizing Compiler
- 1959 : QR Algorithm for Computing Eigenvalues
- 1962 : Quicksort Algorithms for Sorting
- 1965 : Fast Fourier Transform
- 1977 : Integer Relation Detection
- 1987 : Fast Multipole Method

Définition

Un algorithme est une ≪ recette ≫.

Quelques caractéristiques d'une recette :

- toujours dans un état donné,
- un ensemble fini d'entrées (ingrédients),
- un ensemble fini de sorties (plats).

Exemple

Écrivons l'algorithme **pâte feuilletée**.

- Mélanger 200 g de farine et 100 g d'eau
- Étaler sur 1cm d'épaisseur, mettre 200g beurre au milieu. Rabattre les quatre pans de la pâte
- Pour $k = 1$ à 7,
  - Étaler la pâte au rouleau en une bande longue
  - Plier la pâte en 3
  - Tourner la pâte de $\frac{\pi}{2}$
  - Laisser reposer au frais pendant 1 heure
- Renvoyer la pâte.

Exemple

Écrivons l'algorithme **mayonnaise** avec l'entrée c, nombre de convives.

- Mélanger dans un bol $c/4$ jaunes d'œuf, $c/4$ cuillères à soupe de moutarde forte, 1 filet de vinaigre, 1 pincée de sel
- Tant que la mayonnaise n'est pas suffisamment épaisse,
  - Verser un peu d'huile
  - Fouetter
- Renvoyer le bol de mayonnaise

Plus rigoureusement, on énonce la définition

Définition

Un algorithme est une méthode de résolution de problème, décrite par un ensemble fini d'instructions dans un langage formel non ambigu.

Un algorithme est non ambigu mais il peut exploiter une part d'aléatoire.

Précisons une version alternative mais équivalente.

Définition

Un algorithme est une procédure permettant de calculer une fonction numérique.

Exemple

Algorithmes de la somme avec retenues, de la division euclidienne, de la « preuve par 9 », de recherche de racines...

Pour étudier un algorithme, il y a trois points essentiels

Pour étudier un algorithme, il y a trois points essentiels

Terminaison : l'algorithme termine-t-il ?

Pour étudier un algorithme, il y a trois points essentiels

Terminaison : l'algorithme termine-t-il ?

Complexité : en combien de temps (en combien d'opérations) l'algorithme termine-t-il dans le pire des cas ?

Pour étudier un algorithme, il y a trois points essentiels

Terminaison : l'algorithme termine-t-il ?

Complexité : en combien de temps (en combien d'opérations) l'algorithme termine-t-il dans le pire des cas ?

Correction : renvoie-t-il ce que nous attendions ?

Trouver le maximum

Question

Comment trouver le plus grand élément d'un tableau d'entiers ?

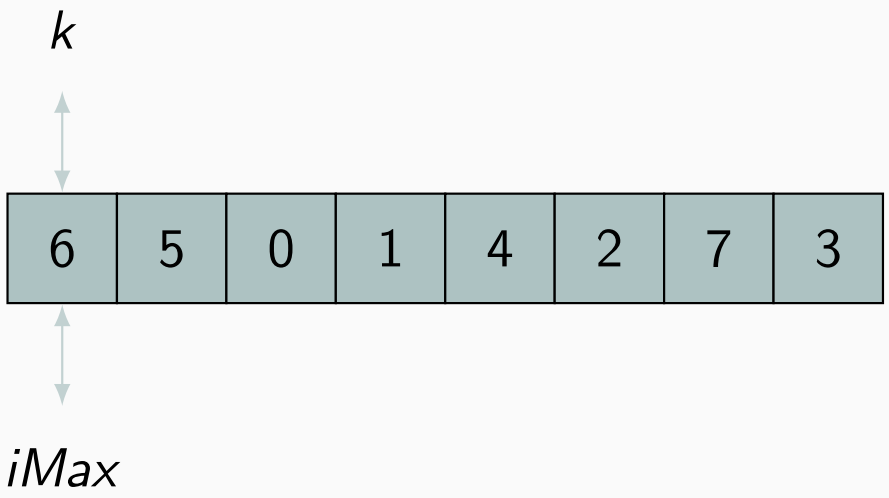
Trouver le maximum

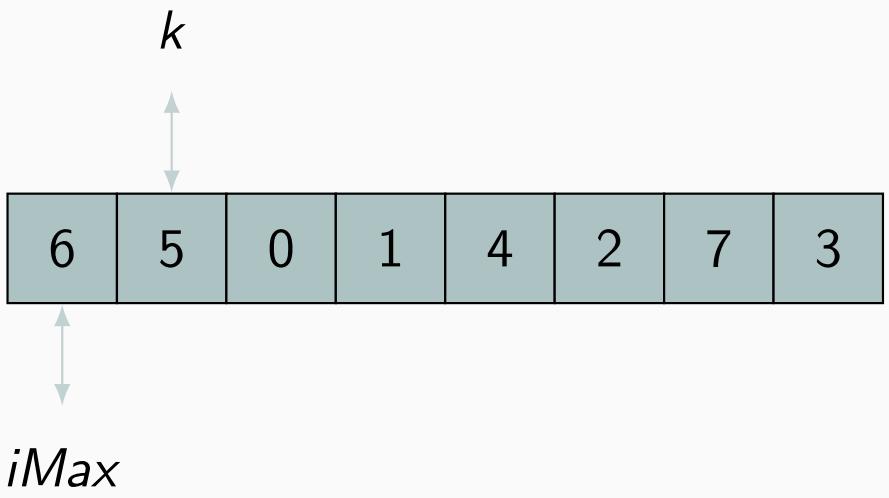
Question

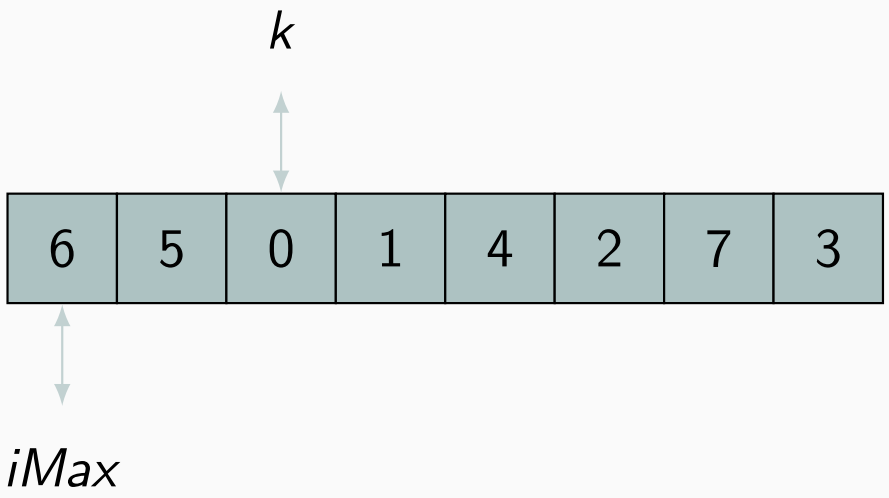
Comment trouver le plus grand élément d'un tableau d'entiers ?

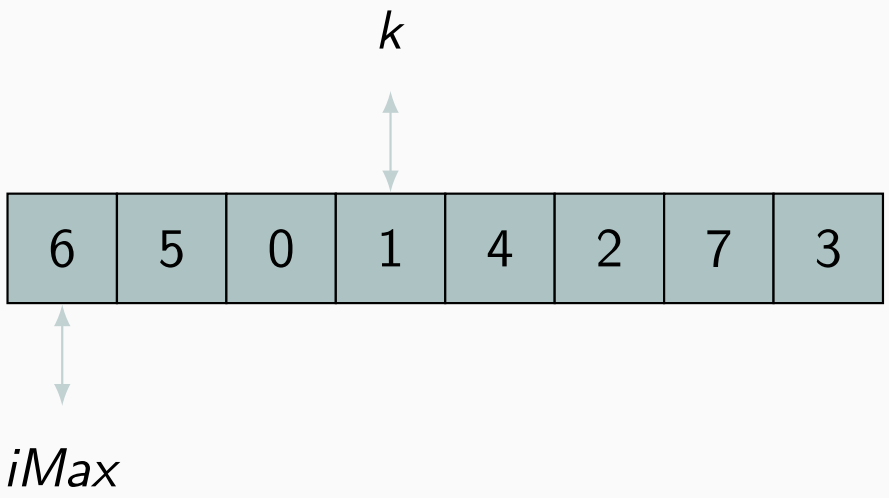
L'idée est de comparer le plus grand déjà obtenu avec le suivant dans le tableau.

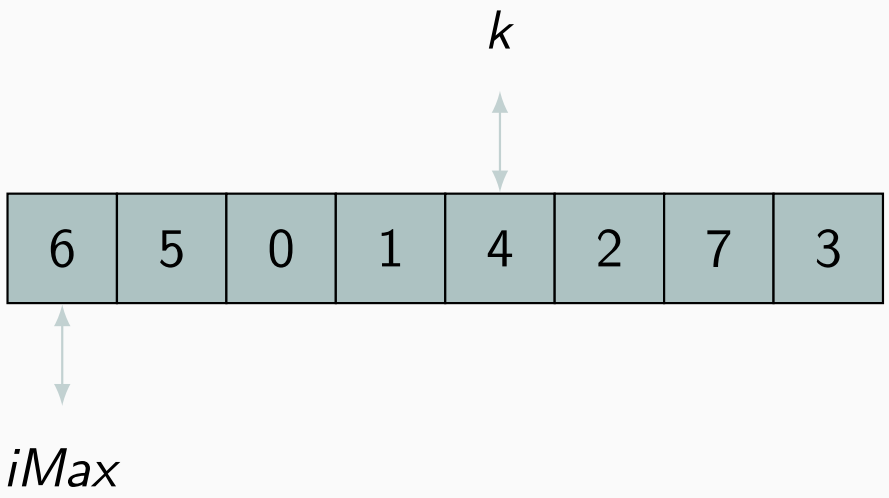
- Initialiser `iMax` à 0
- Pour `k` parcourant (dans l'ordre croissant) les indices du tableau
  - Tester si `t[k]>t[iMax]` et dans ce cas affecter la valeur `k` à la variable `iMax`.
- Renvoyer `t[iMax]`.

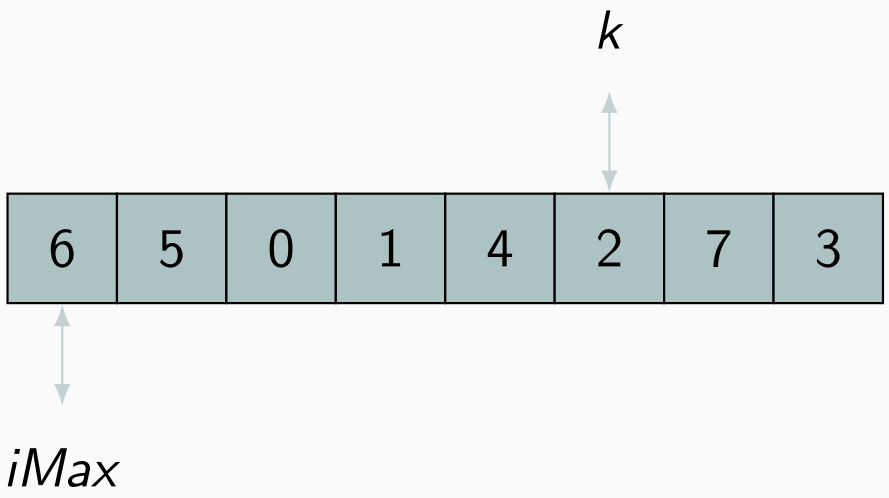

k
6
5
0
1
4
2
7
3
iMax

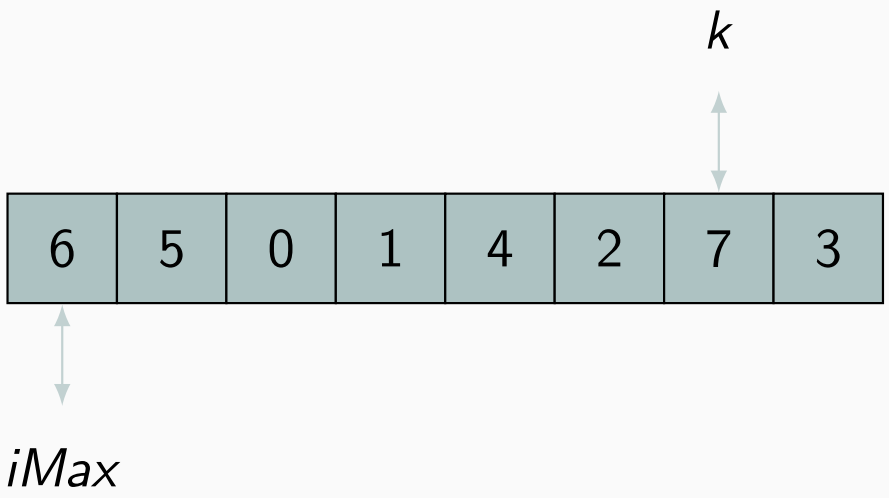

k
6
5
0
1
4
2
7
3
iMax

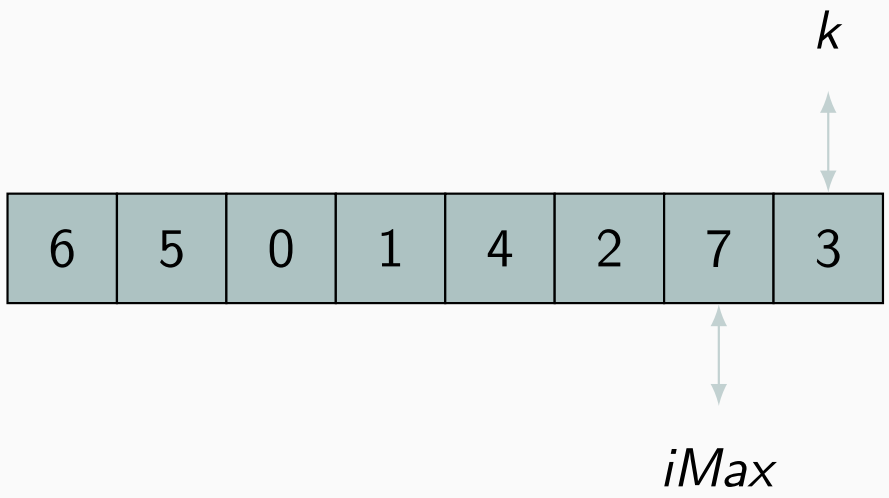

k
6
5
0
1
4
2
7
3
iMax


k
6
5
0
1
4
2
7
3
iMax


k
6
5
0
1
4
2
7
3
iMax


k
6
5
0
1
4
2
7
3
iMax


k
6
5
0
1
4
2
7
3
iMax


k
6
5
0
1
4
2
7
3
iMax

Terminaison : il n'y a qu'un nombre fini de valeurs pour `k` donc un nombre fini d'étapes dans la boucle

Complexité : il y a au plus autant d'étapes que d'éléments dans le tableau

Correction : à chaque étape, `iMax` est l'indice du plus grand élément parmi les `k` premiers ; à la fin, c'est donc l'indice du plus grand élément

Exercice

Comment trouver simultanément le plus grand élément et le plus petit élément d'un tableau d'entiers ?

Tri

Le cas d'école pour la compréhension des algorithmes est le problème du tri d'un ensemble d'éléments par ordre croissant.

Question

Comment trier un tableau d'entiers du plus petit élément jusqu'au plus grand ?

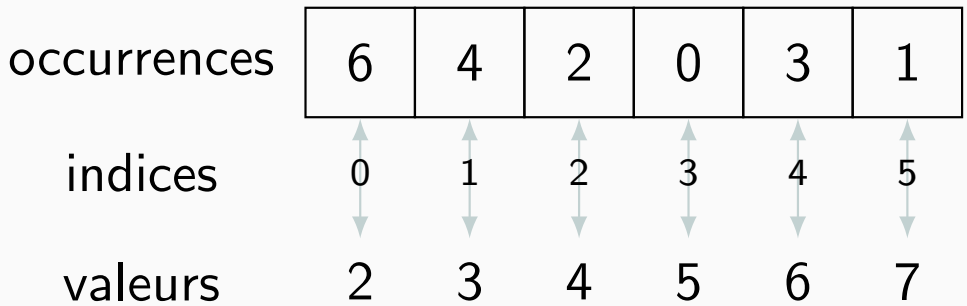
6	5	0	1	4	2	7	3

Tri par comptage

- On parcourt le tableau à la recherche du minimum `m` et du maximum `M` du tableau.
- On parcourt le tableau est on maintient un tableau `o` à `M-m+1` éléments tel que l'élément `o[k]` corresponde au nombre d'occurrences de l'élément `k+m` dans `t`.
- On reconstruit le tableau trié en parcourant `o` en ajoutant `o[k]` fois l'élément `k+m`.

Pour le tableau `t=[2,2,3,4,2,6,2,3,4,6,2,3,7,3,2,6]`, on obtient successivement

- le minimum `m=2` et le maximum `M=7`
- le tableau d'occurrences `o=[6,4,2,0,3,1]`
- le tableau trié `[2,2,2,2,2,2,3,3,3,3,4,4,6,6,6,7]`



La complexité temporelle est linéaire en le maximum des longueurs du tableau initial et du tableau d'occurrences.

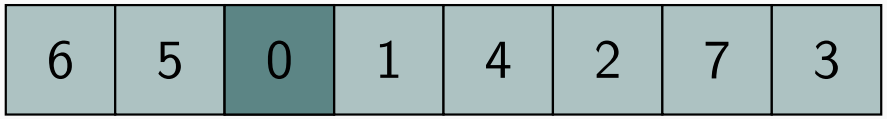
La complexité spatiale est de la taille du tableau d'occurrences.

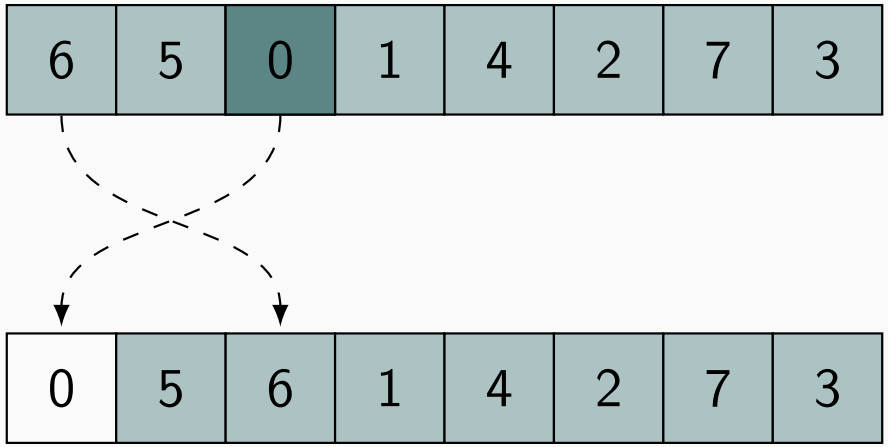
Tri par sélection

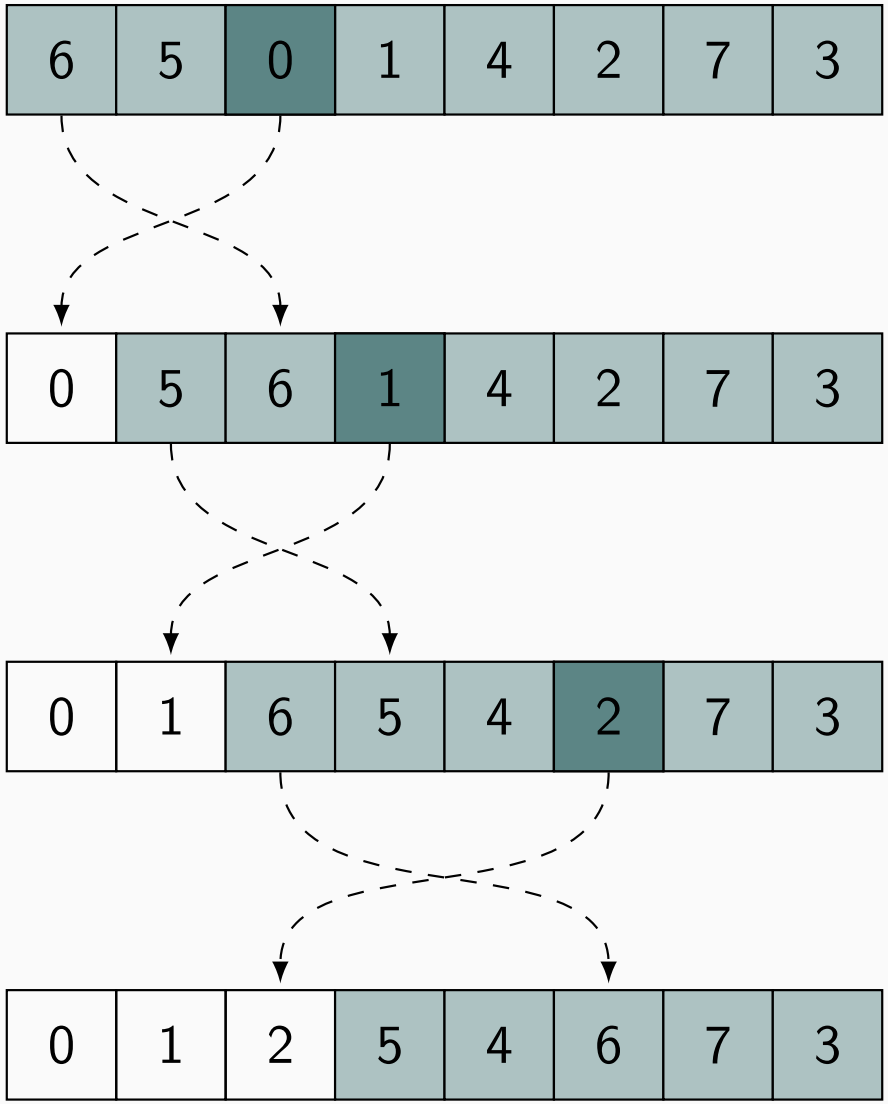
L'idée est de chercher le plus petit élément, de le placer à sa position définitive puis de recommencer avec le reste du tableau.

- Pour `k` parcourant (dans l'ordre croissant) les indices du tableau
  - Chercher l'indice `i` du plus petit élément du tableau entre `k` et la fin.
  - Échanger les éléments d'indices `i` et `k`.
- Renvoyer le tableau.

6	5	0	1	4	2	7	3


6
5
0
1
4
2
7
3


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


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

Invariant de boucle

À l'issue de la `k`-ième étape, les éléments d'indices entre 0 et `k` sont à leur place définitive.

Complexité

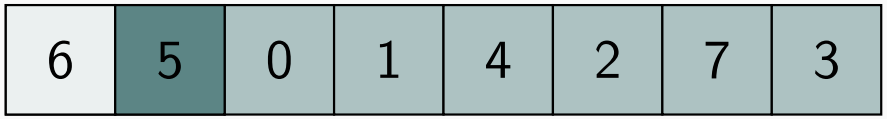
Le nombre de comparaisons pour un tableau de longueur n est

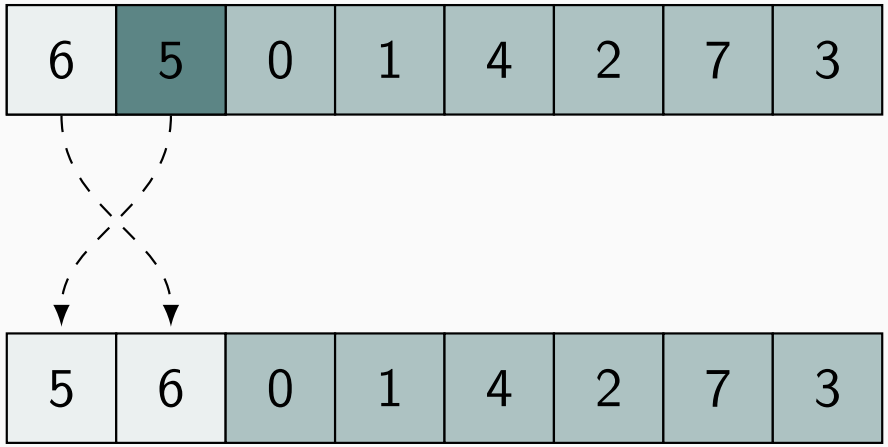
$$\sum_{k=0}^{n-1}(n-k) = \frac{n(n+1)}{2} = O(n^2).$$

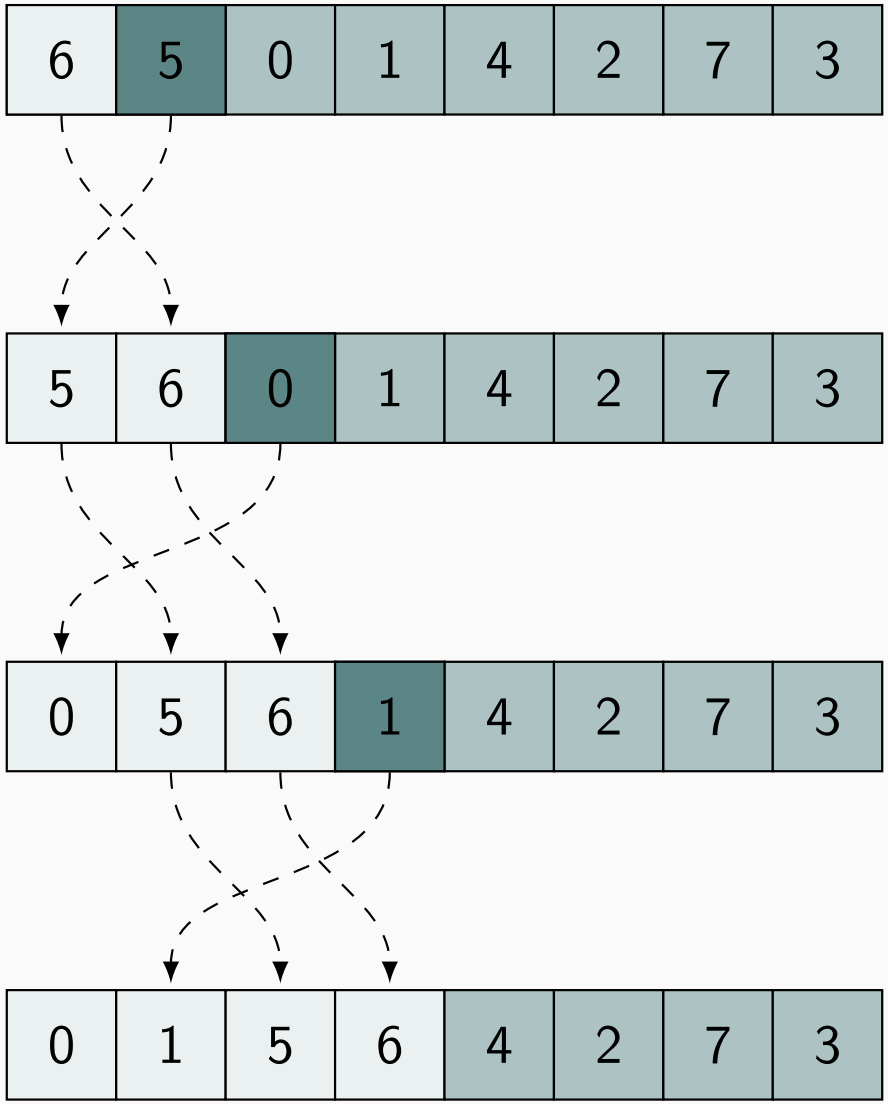
Tri par insertion

L'idée est de prendre les éléments successivement et de les placer à sa position relative par rapport aux éléments déjà positionnés.

- Pour `k` parcourant (dans l'ordre croissant) les indices du tableau, placer l'élément d'indice `k` à la bonne place parmi les éléments d'indices entre 0 et `k-1`.
- Renvoyer le tableau.


6
5
0
1
4
2
7
3


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


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

Invariant de boucle

À l'issue de la `k`-ième étape, les éléments d'indices entre 0 et `k` sont correctement placés les uns par rapport aux autres.

Complexité

Le nombre de comparaisons pour un tableau de longueur n est

- dans le pire des cas (tableau trié à l'envers)

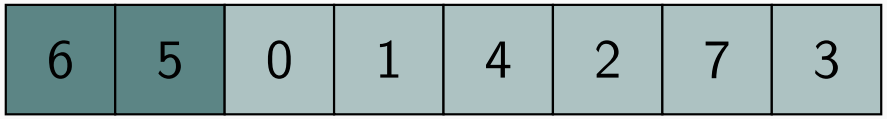
$$\sum_{k=0}^{n-1}(n-k) = \frac{n(n+1)}{2} = O(n^2).$$

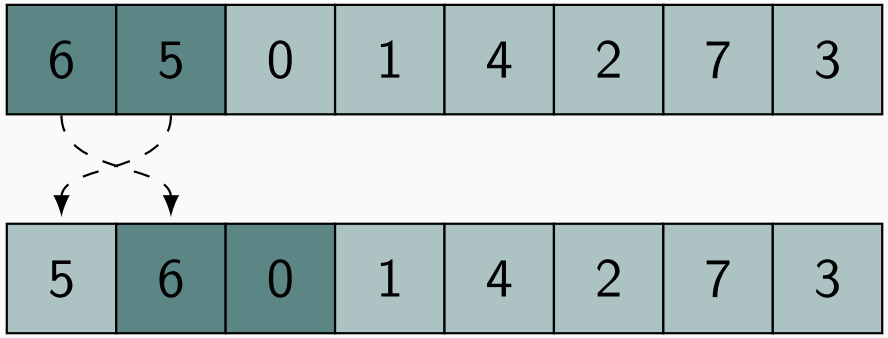
- dans le meilleur des cas (tableau trié dans le bon sens)

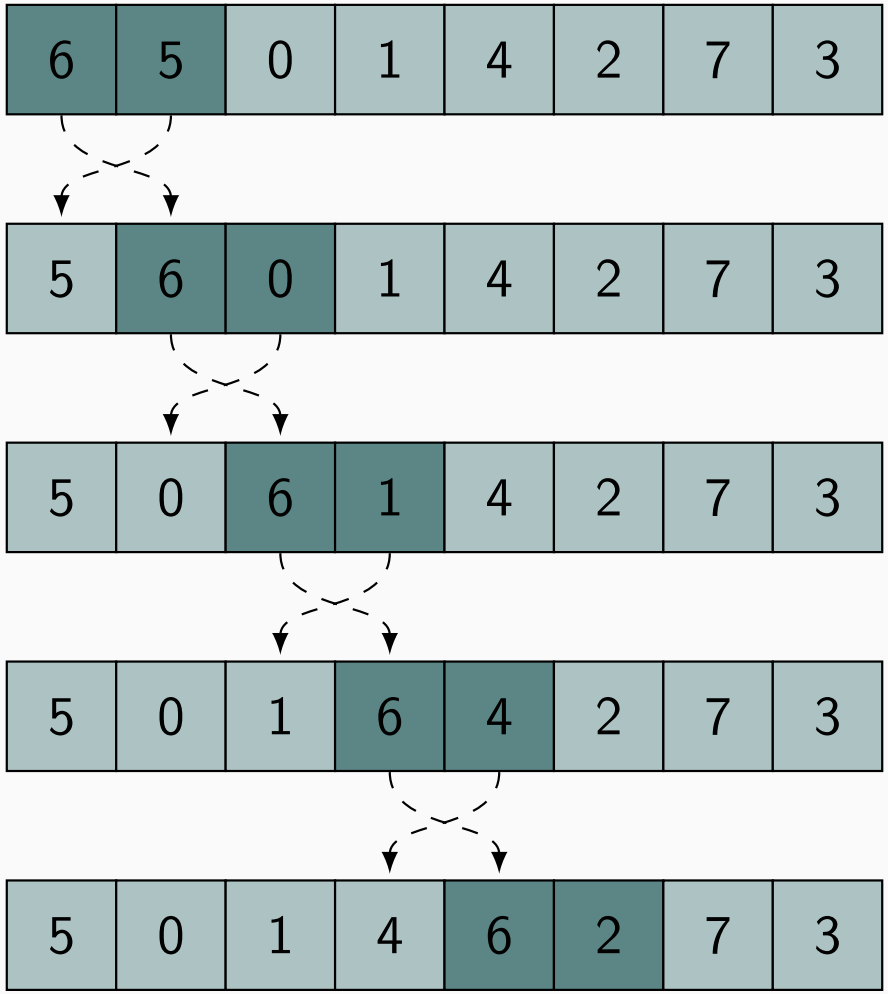
$$\sum_{k=0}^{n-1} 1 = n.$$

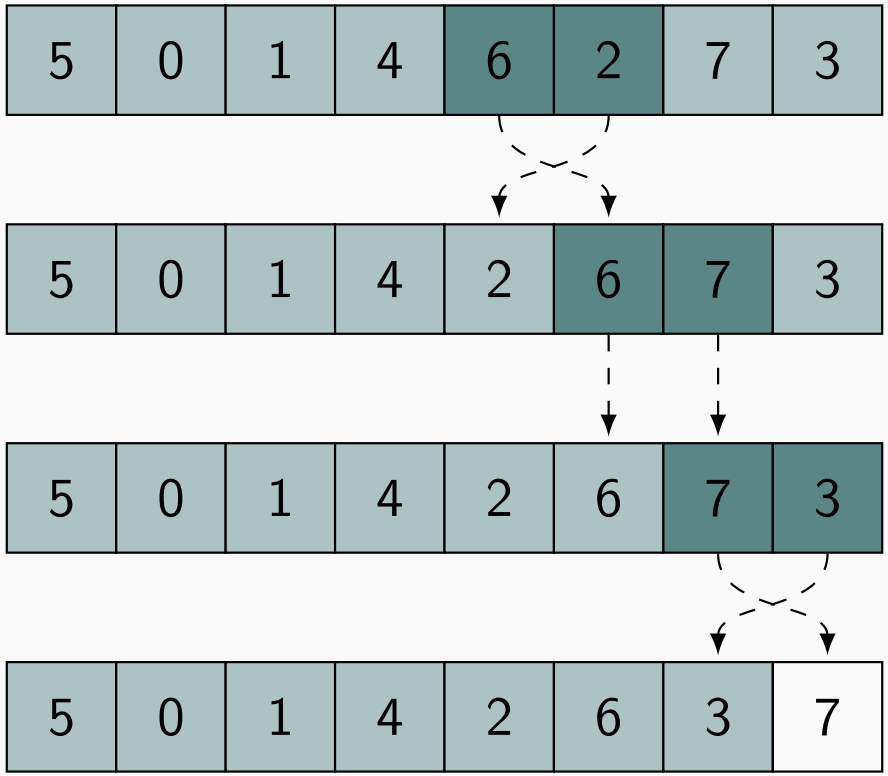
Tri bulle

- Pour `k` parcourant (dans l'ordre croissant) les indices du tableau
  - Pour `j` parcourant (dans l'ordre croissant) les indices du tableau
    on compare les éléments `t[j]` et `t[j+1]`,
    on les échange si besoin.
- Renvoyer le tableau.

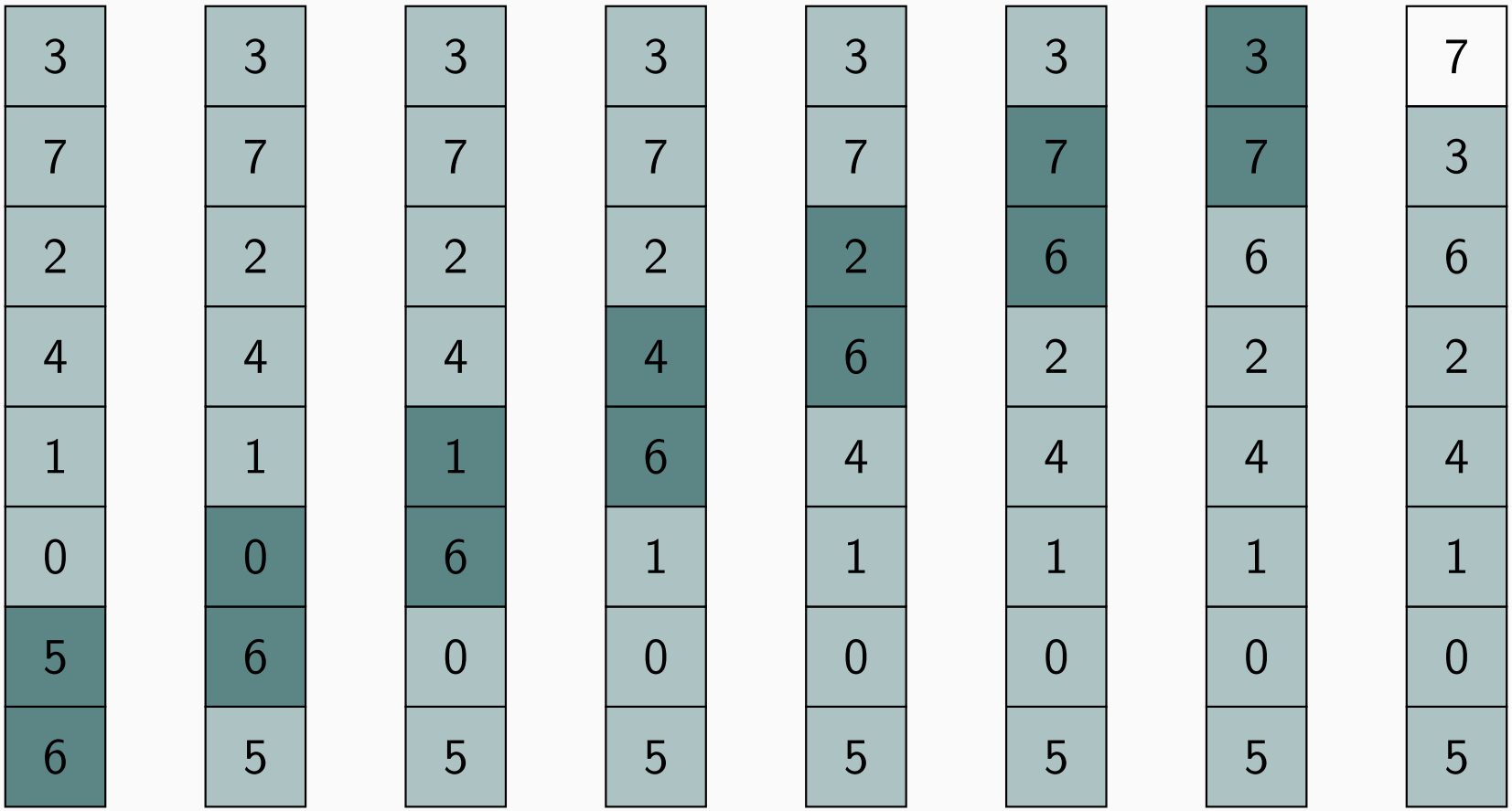

6
5
0
1
4
2
7
3


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


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


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

L'appellation « tri bulle » est plus explicite lorsque l'on représente les tableaux verticalement.



Invariant de boucle

À l'issue de la `k`-ième étape, les `k` plus grands éléments sont à leur place définitive.

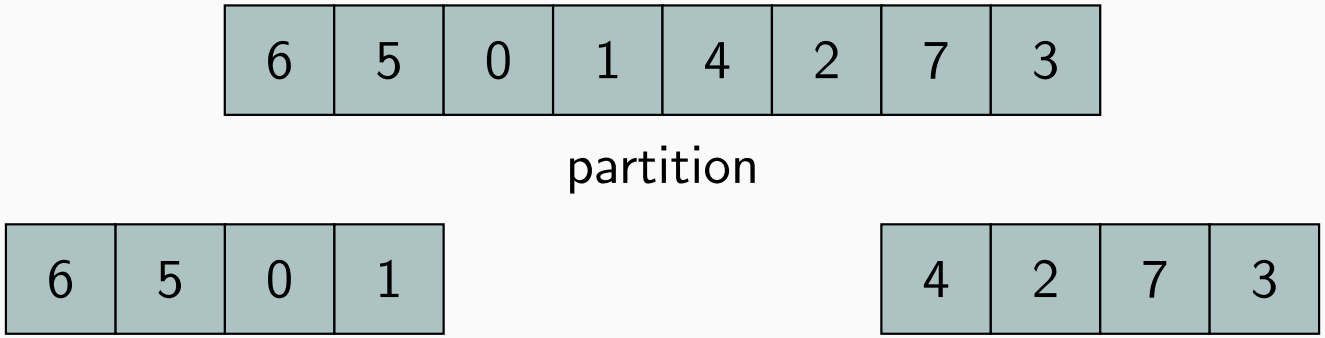
Complexité

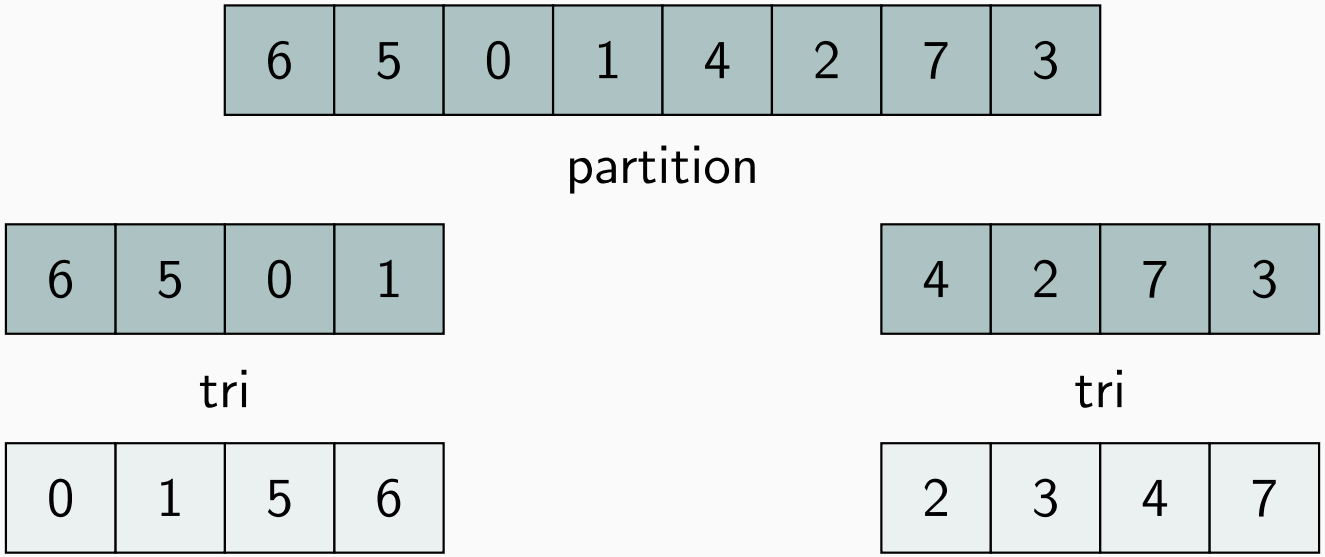
Le nombre de comparaisons pour un tableau de longueur n est

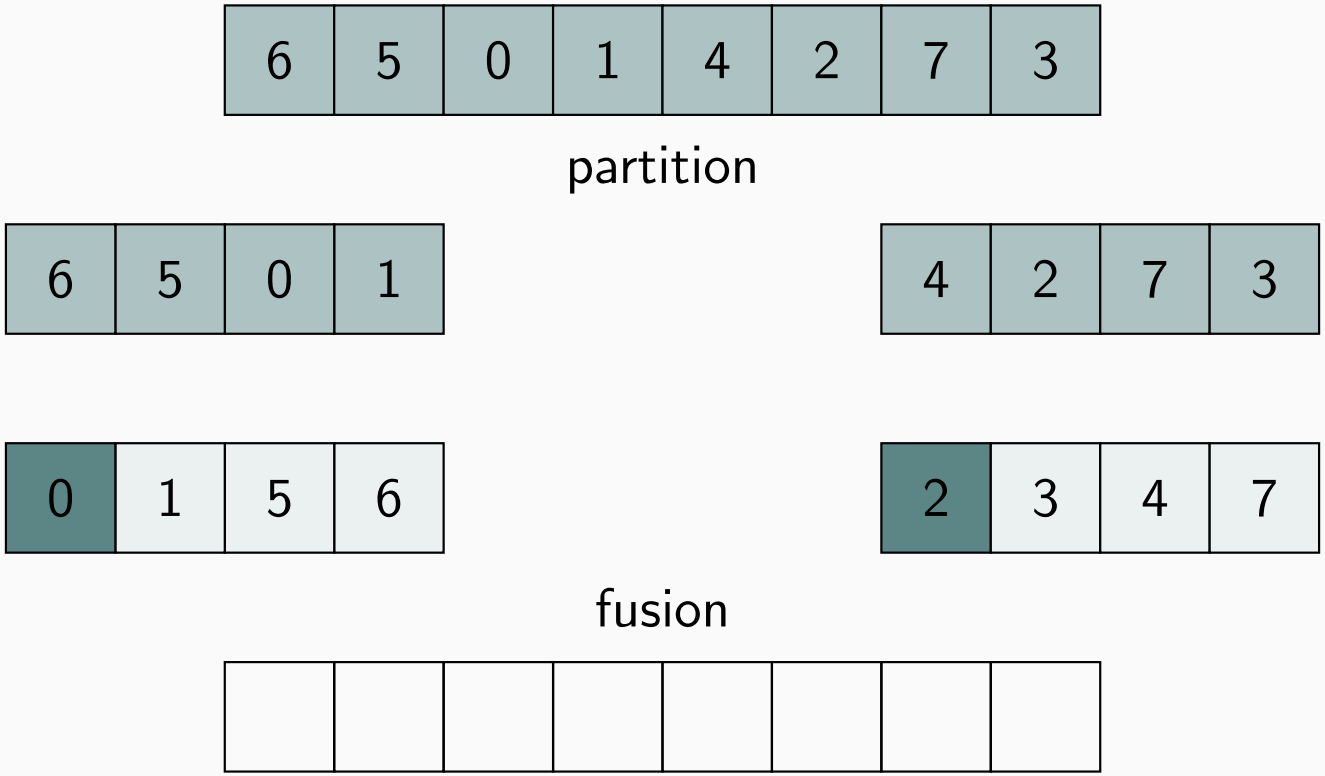
$$\sum_{k=0}^{n-1}(n-1) = n(n-1) = O(n^2).$$

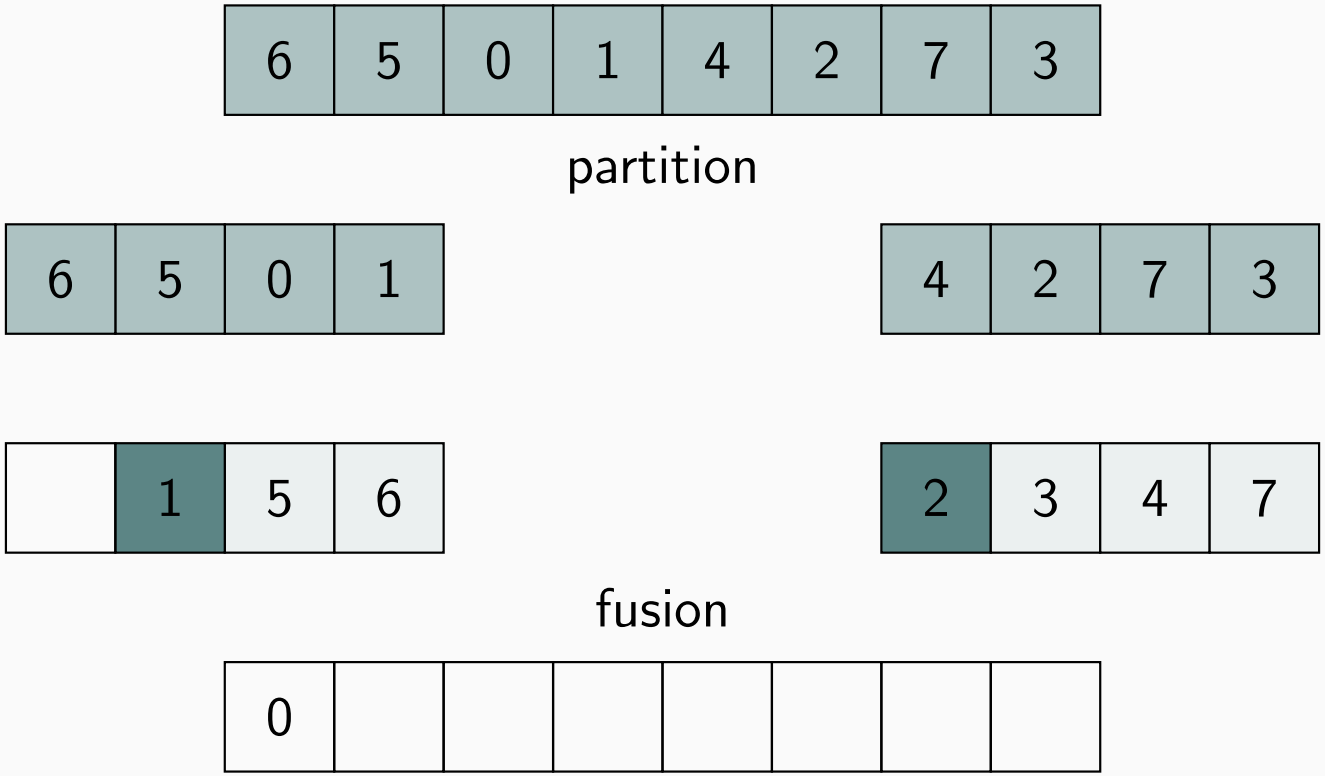
Tri fusion

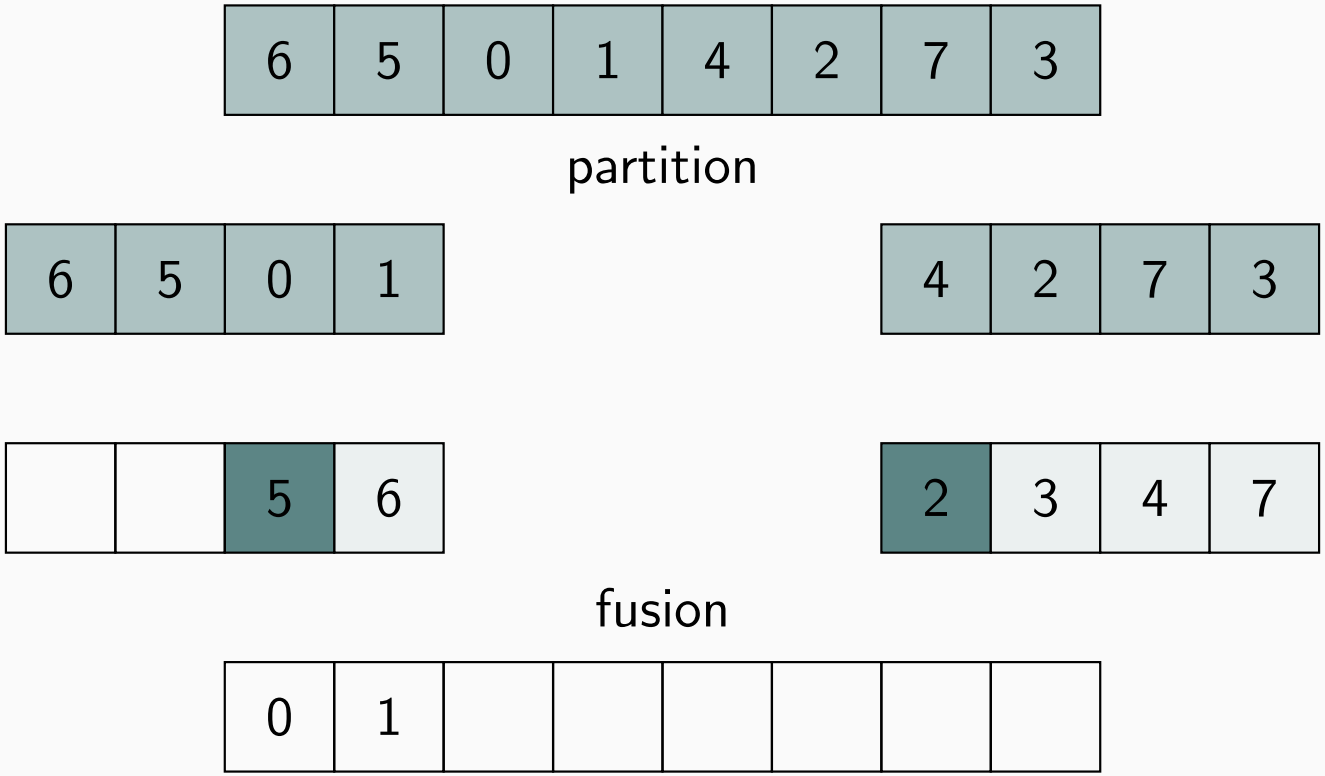
- On sépare le tableau en deux tableaux de tailles « égales ».
- On trie chacun des tableaux
- On fusionne les deux tableaux triés.
  Plus précisément, pour fusionner deux tableaux t_1 et t_2 triés dans l'ordre croissant,
  - On compare les premiers éléments des deux tableaux.
  - On retire le plus petit des deux et on l'ajoute au tableau des résultats
  - On fusionne les tableaux modifiés (par le retrait d'une des deux têtes de tableau).

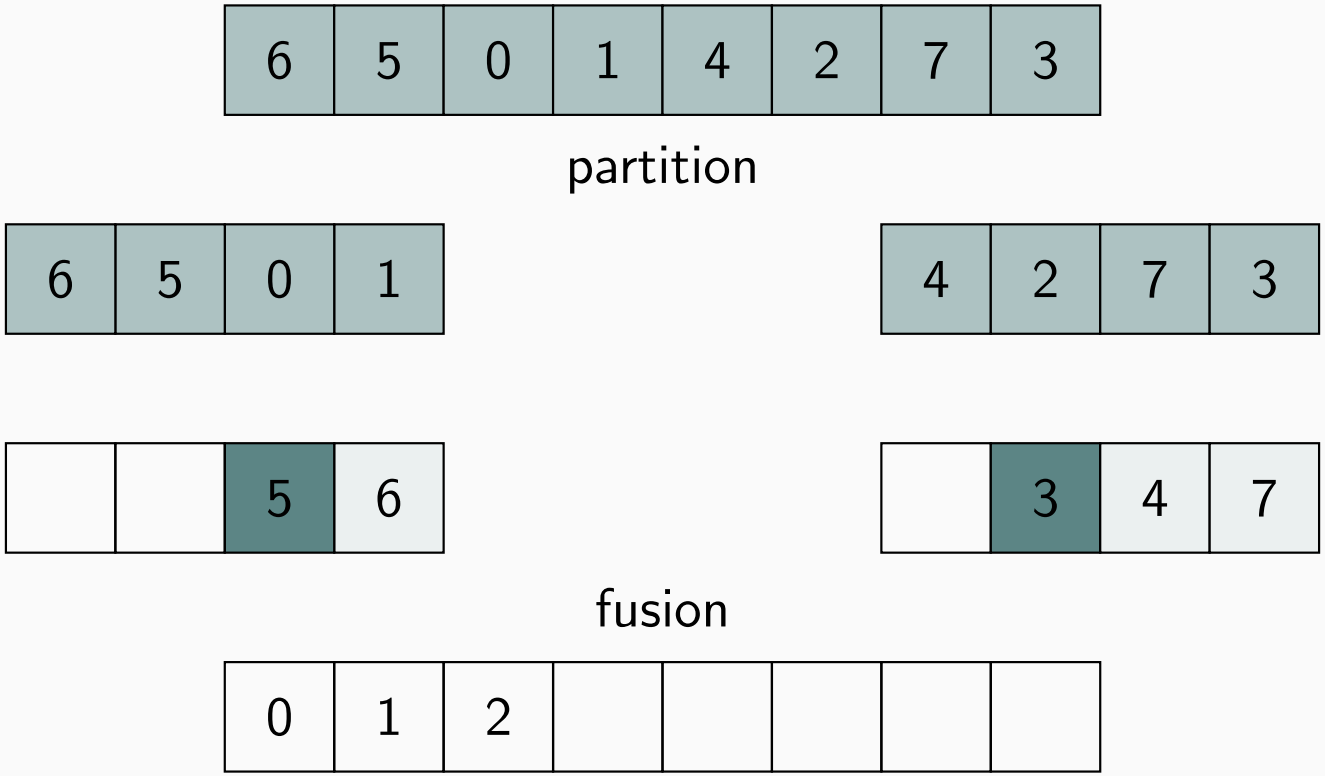

6 5 0 1 4 2 7 3
partition
6 5 0 1
4 2 7 3


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


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


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


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


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

Complexité

Soit u_n le nombre maximal de comparaisons requises pour trier un tableau de longueur n.

Alors,

$$u_n = u_{\lfloor \frac{n}{2} \rfloor} + u_{\lceil \frac{n}{2} \rceil} + n.$$

On peut en déduire que u_n est au plus de l'ordre de $n \log n$.

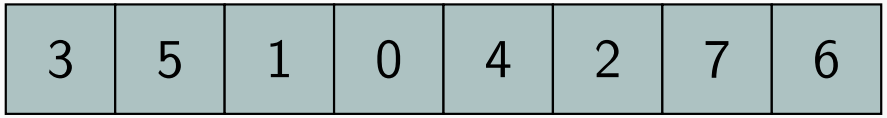
Ce résultat est optimal.

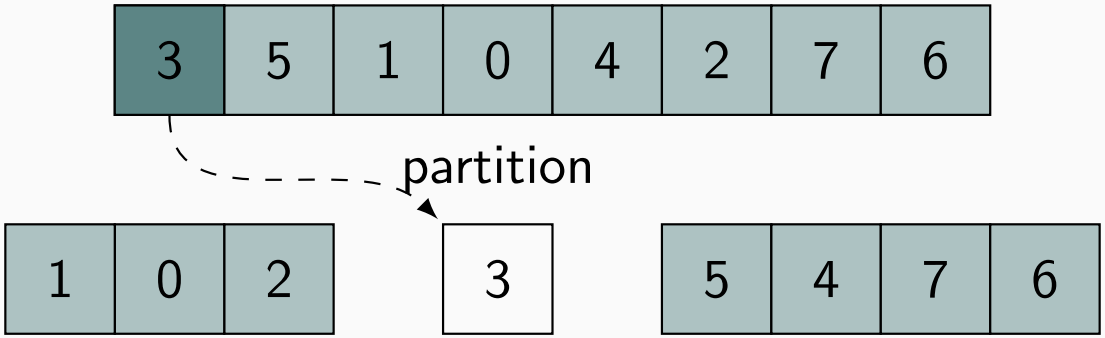
Tri rapide

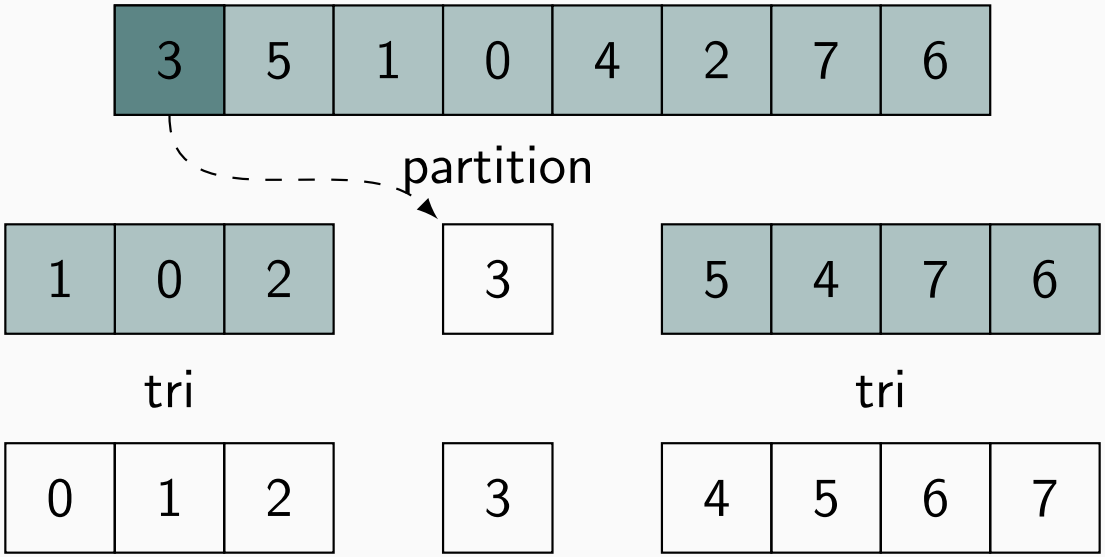
On reprend l'idée du tri dichotomique mais cette fois-ci, on choisit un élément du tableau puis on sépare le tableau en deux par rapport à ce pivot

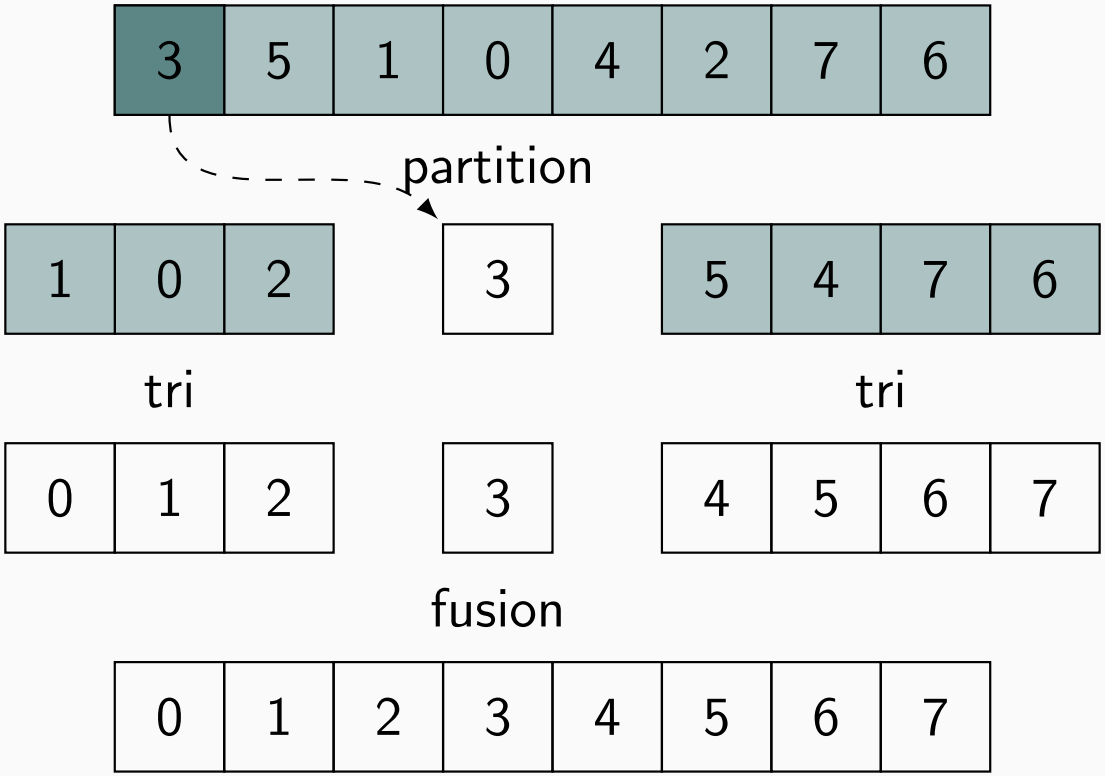
- d'un côté, les éléments plus petits que le pivot,
- de l'autre, les éléments plus grands.

Cette étape de partition réclame un parcours du tableau (alors que pour le tri fusion, la partition était immédiate) mais la fusion est désormais immédiate (simple concaténation).


3
5
1
0
4
2
7
6


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


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


3 5 1 0 4 2 7 6
partition
1 0 2
3
5 4 7 6
tri
tri
0 1 2
3
4 5 6 7
fusion
0 1 2 3 4 5 6 7

Complexité

En moyenne, le nombre de comparaisons requises pour trier un tableau de longueur n est de l'ordre de $n \log n$.

Dans le pire des cas, le nombre de comparaisons requises pour trier un tableau de longueur n est de l'ordre de n^2.

ASKAP : Australian Square Kilometer Array Pathfinder

Ensemble de radiotélescopie ultrasensible inauguré en 2012 (collaboration Australie, Canada, États-Unis, Pays-Bas).

Il fournit une ≪ image ≫ numérique du ciel.

Pour traiter les nombreuses images ainsi obtenues, il faut avoir recours à une procédure automatisée.

Question

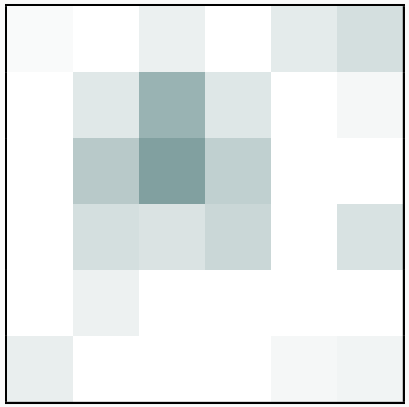
Comment repérer automatiquement la zone d'intérêt (ou les zones) dans une image numérique ?

En première approximation, une image numérique est un tableau de valeurs (intensité lors d'observations de plusieurs heures à une fréquence donnée dans le cas de la radiotélescopie).

Par exemple,

3	0	10	0	13	21
0	15	50	16	0	5
0	35	62	31	0	0
0	21	18	26	0	19
0	9	0	0	0	0
11	0	0	0	5	7

correspond à



Dans cet exemple, on cherche sûrement à isoler

3	0	10	0	13	21
0	15	50	16	0	5
0	35	62	31	0	0
0	21	18	26	0	19
0	9	0	0	0	0
11	0	0	0	5	7

On regroupe l'ensemble des techniques de recherche d'un élément homogène (contraste, luminosité...) significatif dans une image sous le nom de *détection de blob* ; Ceci requiert beaucoup de géométrie différentielle.

D'autres idées reposent sur la détection de contours, de coins...

Max Subarray

Précisons le problème

Question

Comment, dans un tableau de nombres, repérer le sous-tableau, c'est-à-dire un rectangle, dont la somme des valeurs est maximale ?

Exercice

Essayer avec

-1	2	-3	5	-4	-8	3	-3
2	-4	-6	-8	2	-5	4	1
3	-2	9	-9	3	6	-5	2
1	-3	5	-7	8	-2	2	-6

L'idée naïve consiste à être exhaustif : on calcule toutes les sommes associées à tous les sous-tableaux et on détermine le maximum de ces sommes.

Si on suppose que le tableau est de taille $n \times n$, il y a pour cet algorithme

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=i}^{n}\sum_{l=j}^{n}(k-i)(l-j) = O(n^6)$$

additions.

Pour comprendre qu'il y a beaucoup trop d'opérations ici, considérons l'algorithme de Kadane (1984) suivant pour résoudre le problème avec un tableau à une dimension

- Initialiser à 0 des variables stockant la somme maximale déjà rencontrée `max`, la somme en cours `maxCourant`
- Pour `k` parcourant (dans l'ordre croissant) les indices du tableau
  - Calculer `temp` comme `maxCourant` + `t[k]`
  - Si `temp` $<$ 0, remplacer
    - `maxCourant` par 0.
  - Si `temp` $>$ `max`, remplacer `max` par `temp`
  - Remplacer `maxCourant` par `temp`
- Renvoyer `max`.

Invariant de boucle

À l'issue de la `k`-ième étape, `max` désigne la plus grande somme obtenue avec un sous-tableau de la partie du tableau de départ composée de k éléments.

Complexité

Le nombre d'additions pour un tableau de longueur n est

$$\sum_{k=0}^{n-1} 1 = n.$$

On peut également introduire des variables correspondant aux indices de ces sommes et les actualiser quand nécessaire.

- Initialiser à 0 des variables stockant la somme maximale déjà rencontrée `max`, la somme en cours `maxCourant`, les indices `d` et `f` de début et fin de la somme maximale, l'indice `dC` du début de la somme courante.
- Pour `k` parcourant (dans l'ordre croissant) les indices de la liste
  - Calculer `temp` comme `maxCourant` + `l[k]`
  - Si `temp` $<$ 0, remplacer
    - `maxCourant` par 0
    - `dC` par `k+1`.
  - Si `temp` $>$ `max`, remplacer
    - `max` par `temp`
    - `d` par `dC`
    - `f` par `k`.
  - Remplacer `maxCourant` par `temp`
- Renvoyer `d` et `f`.

Pour un tableau de taille $n \times 1$, cet algorithme fait au plus n additions.

La version naïve en ferait de l'ordre de n^3.

Pour un tableau de taille $n \times 1$, cet algorithme fait au plus n additions.

La version naïve en ferait de l'ordre de n^3.

Malheureusement, ceci ne se généralise pas pour un tableau à deux dimensions.

Takaoka (2002) a l'idée est de recourir à nouveau au paradigme diviser pour régner en séparant le tableau en deux parties droite et gauche.

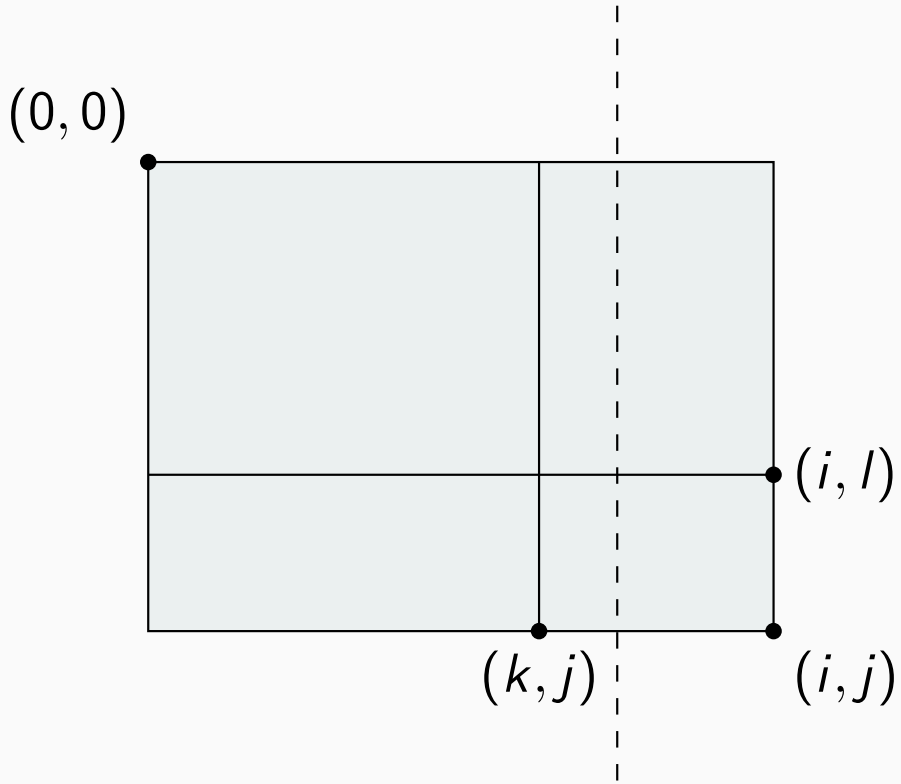
- On calcule la somme maximale à gauche S_g.
- On calcule la somme maximale à droite S_d.
- On calcule la somme maximale franchissant la colonne centrale S_c.

On renvoie $\max(S_g, S_d, S_c)$.

Il y a évidemment une difficulté pour calculer S_c.

Pour cela, on calcule toutes les sommes $S_{i,j}$ sur la partie du tableau de coin haut-gauche $(0,0)$ et de coin bas-droit (i,j).

Puis on obtient $S_c = \max_{i,j,k,l}\{S_{i,j} - S_{i,l} - S_{k,j} + S_{k,l}\}$



Cette dernière formule se ramène à un calcul de deux produits matriciels (dans les algèbres $(\max, +)$ et $(\min, +)$) via la transformation

$$\sum_k a_{i,k} b_{k,j} \qquad \rightarrow \qquad \max_k (a_{i,k} + b_{k,j}).$$

Ce problème est alors bien connu et des algorithmes rapides sont documentés.

Proposition

Pour une matrice de taille $n \times n$, la complexité de cet algorithme

$$O(n^3 \sqrt{\frac{\log \log n}{\log n}}).$$

Cette version simple d'un algorithme de « traitement d'images » a été adaptée (avec la parallélisation) pour SKA.

Elle est également exploitée en oncologie.

De nouveaux algorithmes sont inventés en permanence pour des questions de détection : en juillet 2016, une équipe du MIT a présenté un algorithme *Continuous High-resolution Image Reconstruction using Patch priors* plus adapté à la recherche de trous noirs à partir des données de *Event Horizon Telescope*.

Merci