Informatique 1

10. Tris et complexité

Dr Pierre-André Mudry
pierre-andre.mudry@hevs.ch

Rev 1.51

Une méthode de résolution de problèmes

1. ALGORITHMES

Introduction

- *Collections* : stocker information, dynamique.
- Information brute est peu utile… il faut pouvoir ***trier*** et faire des ***recherche*** dans cette information
- Exemples :

Comment trier ?

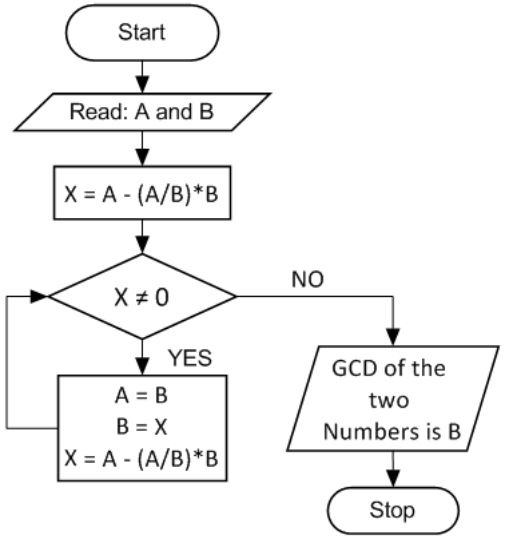
```
int[] a = {4, 7, 1, 3, 6}
```

Algorithme

Ensemble des étapes pour résoudre un problème en utilisant un nombre fini d'instructions.

Algorithme

a)



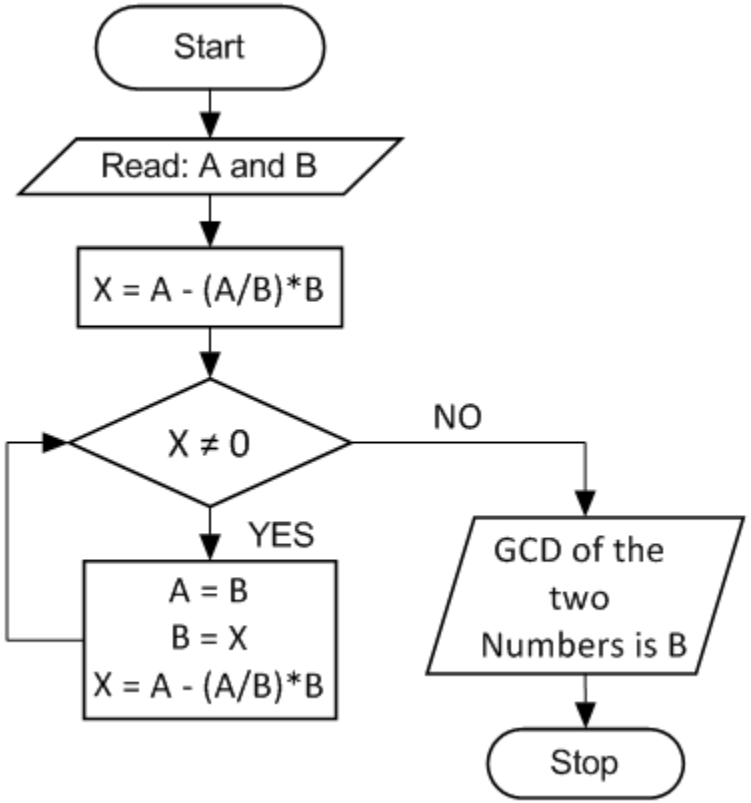
b)

```
if <condition> do stuff;
else do other stuff;

while <condition> do stuff;

...
```

Exemple : le plus grand diviseur commun



Quelques notions

2. COMPLEXITÉ ALGORITHMIQUE

Idée de complexité

- *Tautologie* : plus problème est grand, plus ça prendra de temps pour le résoudre

Idée de complexité (2)

Tri de 10k éléments : *1.67 secondes*

x2

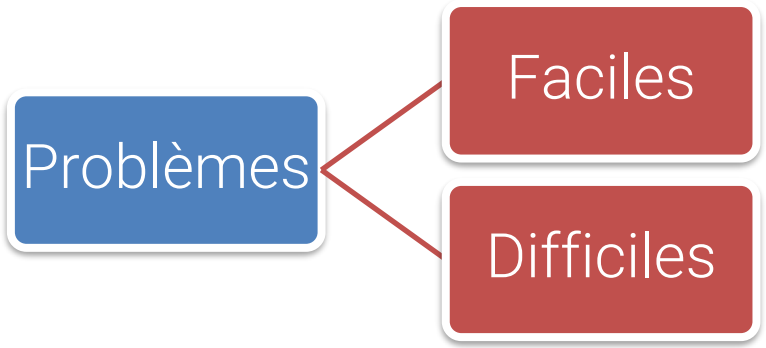
Tri de 20k éléments :

- *Question :* combien de fois plus de temps quand on augmente n ? → *complexité*

La notion de complexité

Lien entre taille *n* d'un problème et temps / mémoire pour le résoudre

Complexité



Tous les algorithmes ne sont pas équivalents

Algorithmique (1)

Branche des mathématiques
/ informatique théorique

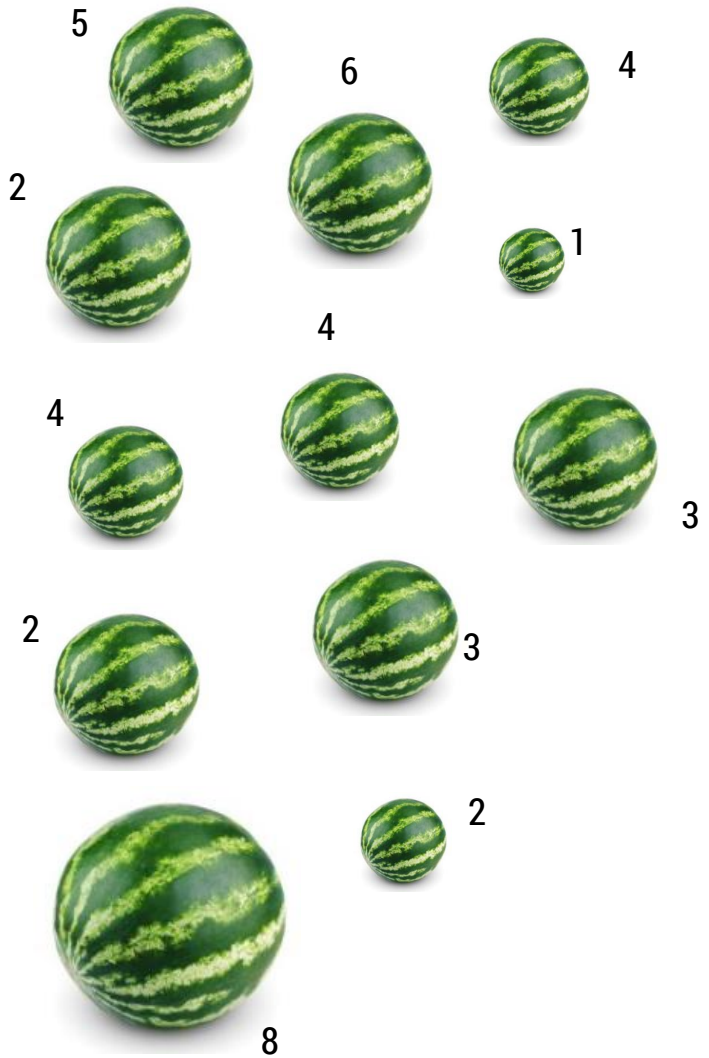
Analyse des algorithmes

Problèmes difficiles $\in$ NP

Problèmes faciles $\in$ P

Théorème de la NP-complétude (P = NP ?)

Exemple NP : watermelons



12 kg max

Solution possible

5, 6

4, 4, 4

2, 3, 3, 1

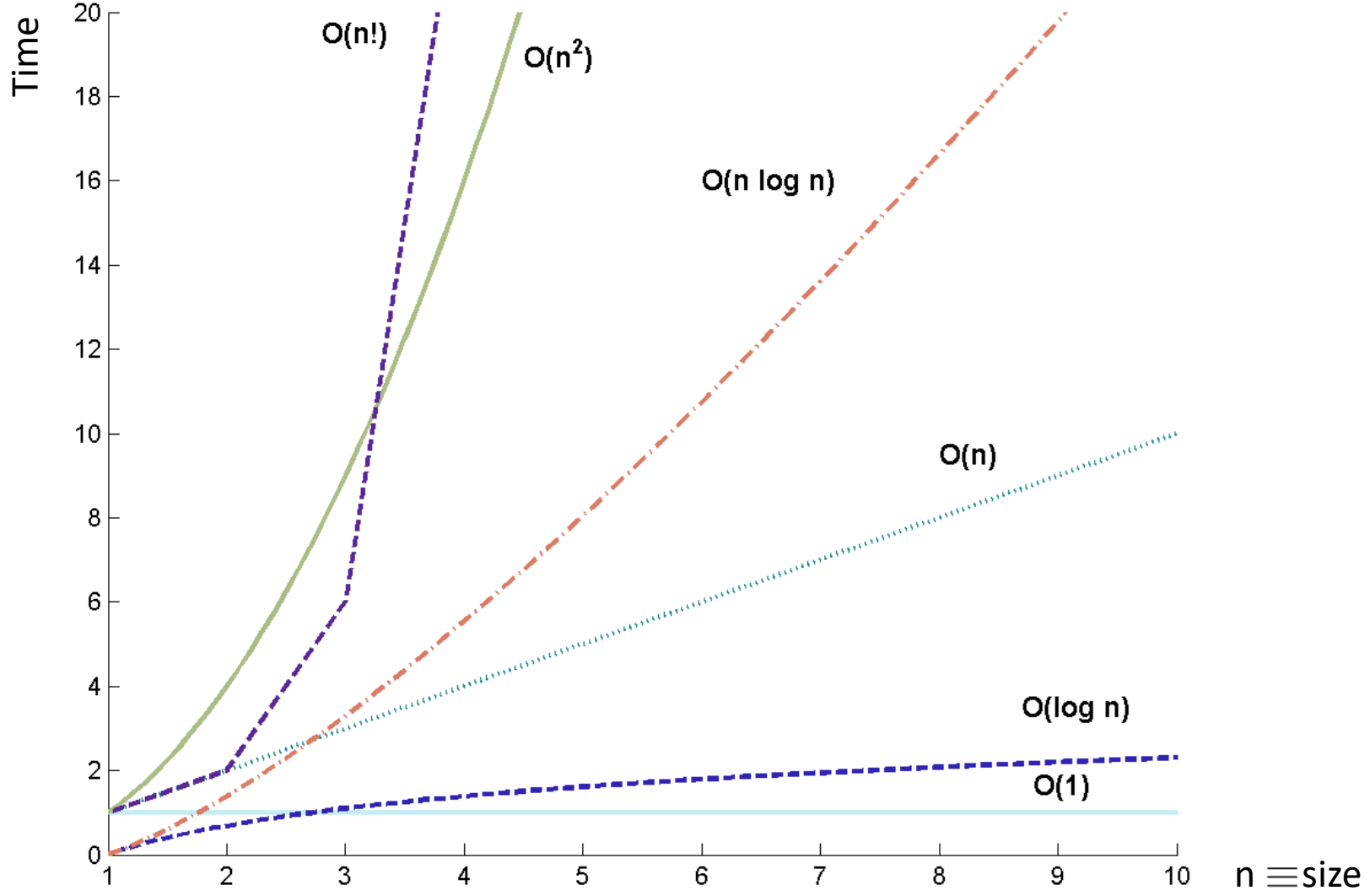
8, 2, 2

Algorithmique (2)

- En algorithmique, notation $\mathcal{O}()$
- Nommé ***ordre*** de la complexité. Obtenu si taille n du problème tend vers $+\infty$
- *Exemples* :

$$\mathcal{O}(1), \mathcal{O}(n), \mathcal{O}(n^2), \mathcal{O}(nlog(n))$$

La complexité en images

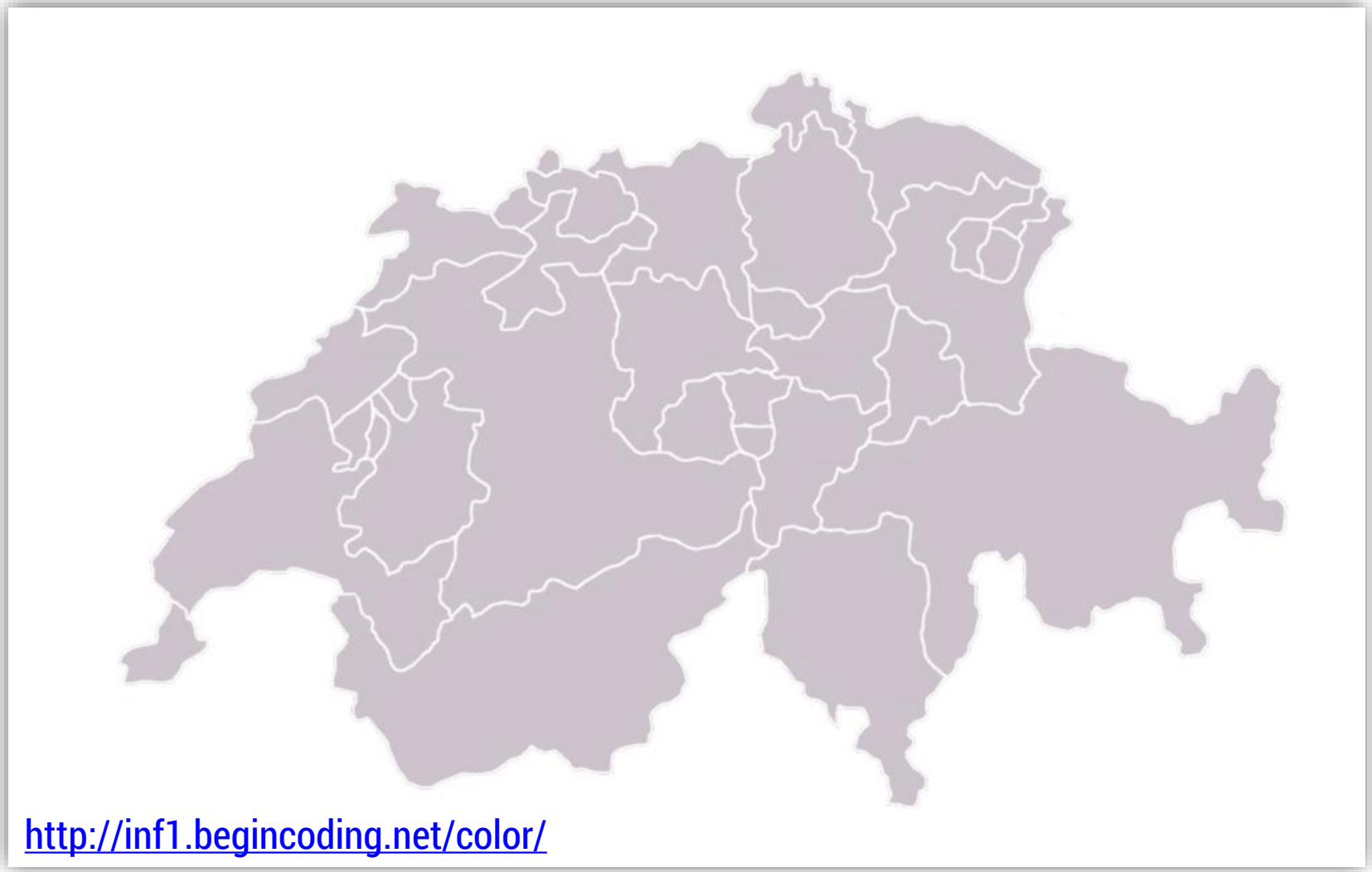


La complexité en chiffres

n	n^2	$n!$
1	1	1
2	4	2
3	9	6
4	16	24
5	25	120
6	36	720
7	49	5040
8	64	40'320
9	81	362'880
10	100	3'628'800

Handshakes

Four color theorem



http://inf1.begincoding.net/color/

Explications https://www.youtube.com/watch?v=g_nTfZ9OgJs

Quiz

Arranger selon un ordre

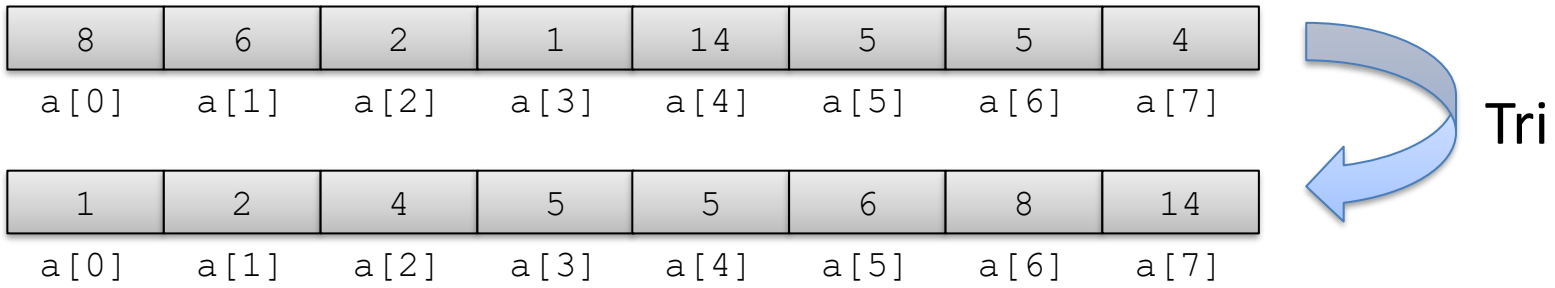
3. ALGORITHMES DE TRIS

Algorithmes de tri

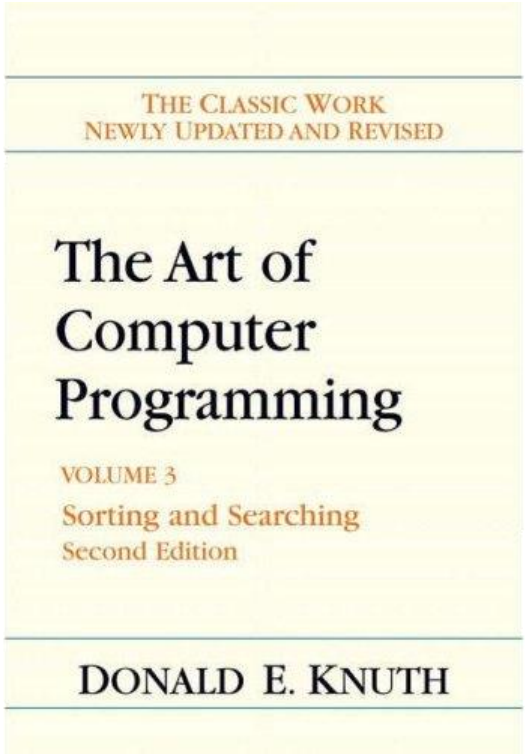
Une méthode de tri prend une structure de donnée **a** et réarrange les éléments de **a** afin que, lorsque la méthode termine, les éléments de **a** soient triés dans l'ordre croissant.

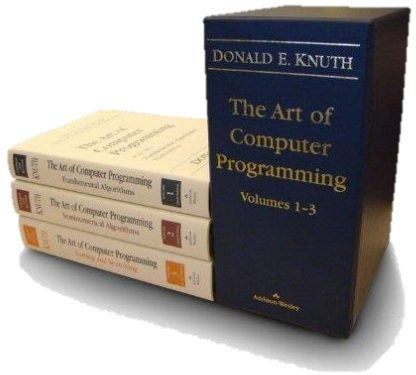
Méthode de tri

Autrement dit :

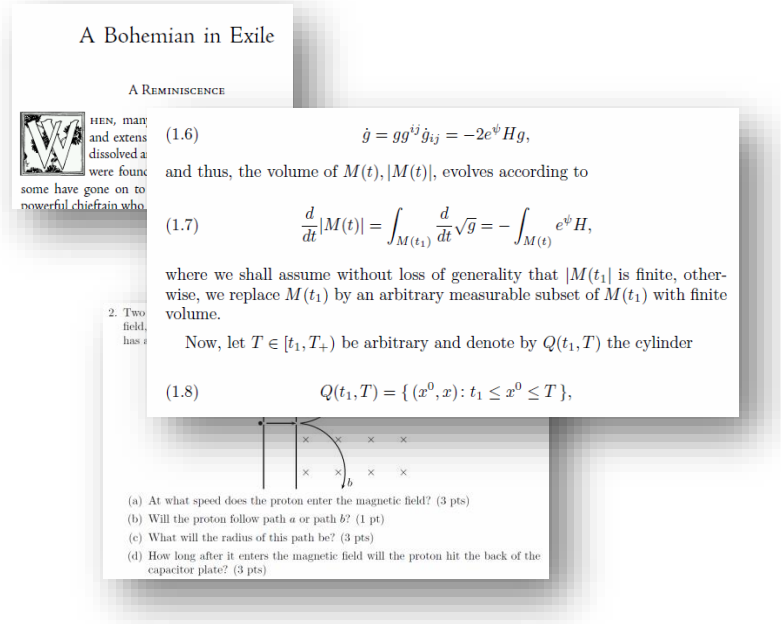


Donald E. Knuth





LaTeX



Tri par sélection

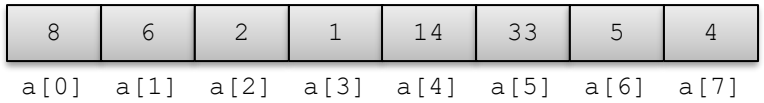
- Pseudo-code pour un tableau `a` à trier:

```
// Selection sort pseudo-code
for (i = 0; i < size[a]; i++){
   Mettre le i-ème plus petit élément dans a[i]
}
```

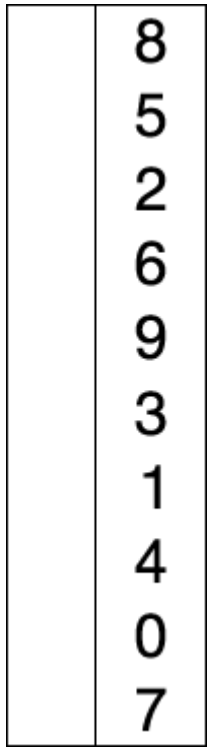
- Code complet à réaliser au labo…

Tri par sélection

Initial array



Tri par sélection, autre exemple



Tri par bulles (bubble sort)

- *Idée* : regarder des paires d'éléments adjacents du début à la fin et les échanger si pas dans l'ordre. On recommence tant que pas trié.

```
// Bubble sort pseudo-code
Tant que a n'est pas trié
   for(i= 0; i< a.length-1; i++)
      if(a[i] > a[i+1])
         Echanger a[i] et a[i+1]
```

En vidéo

Complexité des tris examinés

- Algorithmes vus sont les plus simples mais les plus performants
- Il en existe beaucoup d'autres

  http://www.sorting-algorithms.com/ ou http://sorting.at/#
- Résumé complexité quelques algorithmes :

Methode	Best case	Average case	Worst case
Insertionsort	$\mathcal{O}(n)$ Si déjà trié	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$ Ordre inversé
Selectionsort	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Bubblesort	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Quicksort	$\mathcal{O}(nlog(n))$	$\mathcal{O}(nlog(n))$	$\mathcal{O}(n^2)$
Mergesort	$\mathcal{O}(nlog(n))$	$\mathcal{O}(nlog(n))$	$\mathcal{O}(nlog(n))$

Conclusion

- Complexité → très important car mauvais choix d'algorithme = mauvais programme.
- Ordinateur rapide pas suffisant :