



Informatique 1

10. Tris et complexité

Objectifs

Trier des données

- ▶ Algorithmes
- ▶ Complexité algorithmiques
- ▶ Quelques algorithmes de tri

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 ?

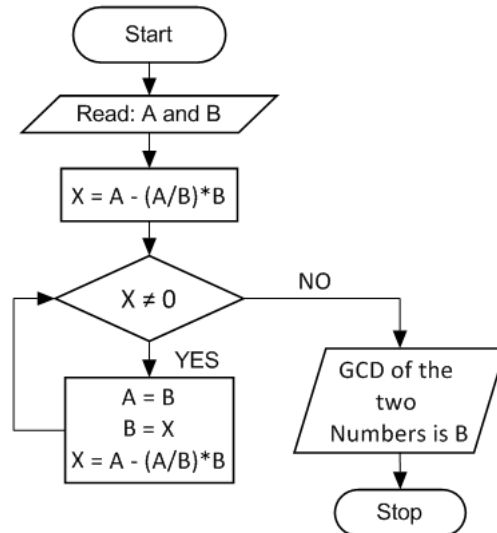
```
int[] a = {4, 7, 1, 3, 6}
```

Algorithme

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

Algorithme

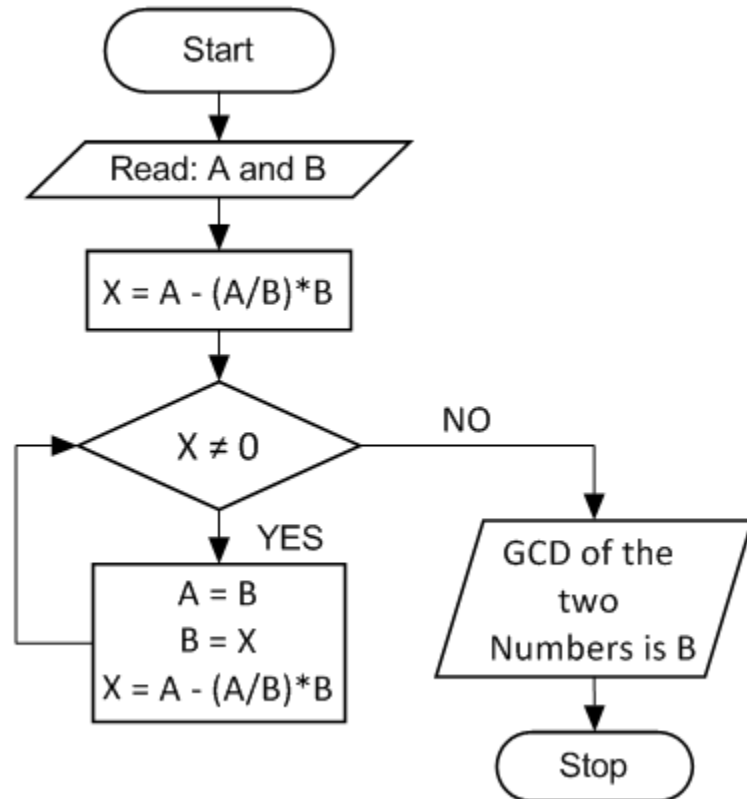
a)



b)

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

Exemple : le plus grand diviseur commun

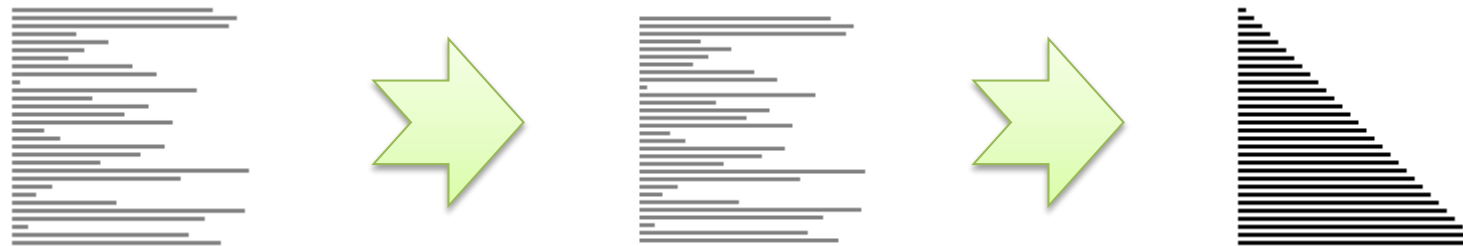


Quelques notions

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



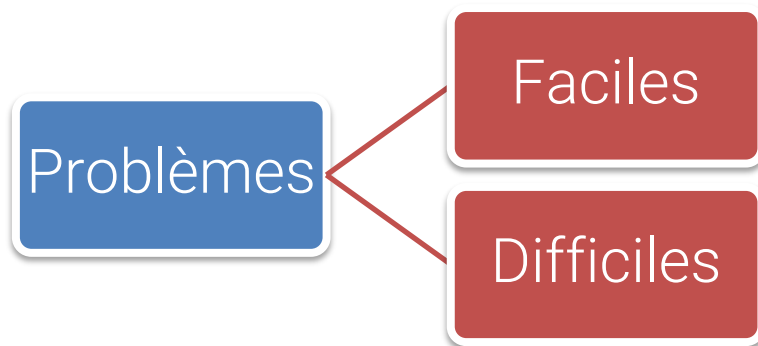
Tri de 20k éléments :

- *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

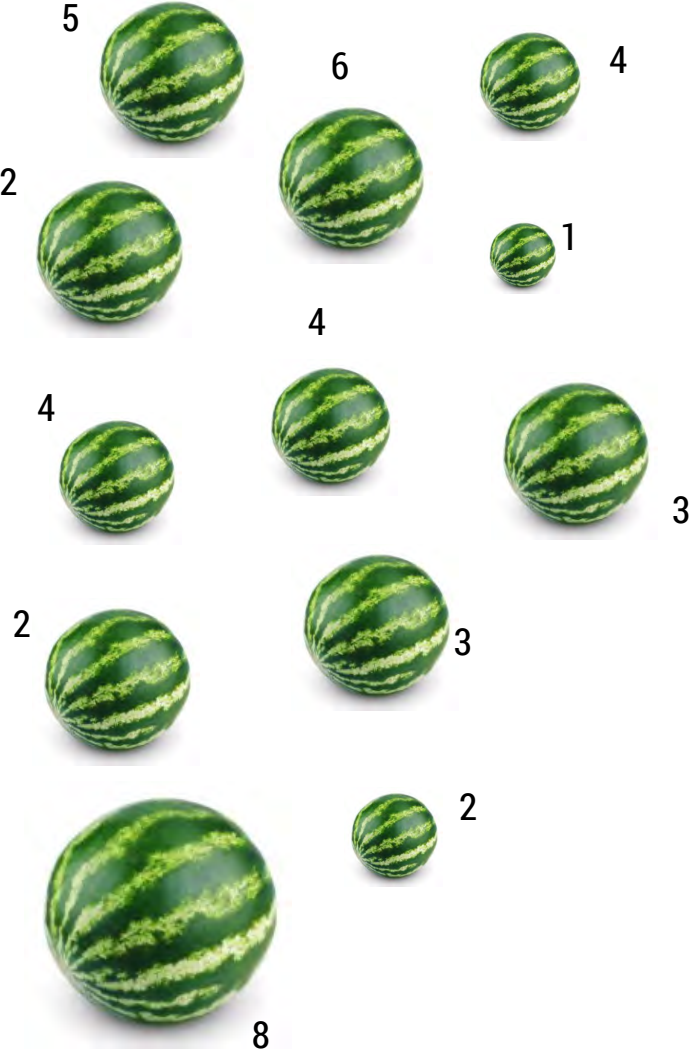
Analyse des algorithmes

Problèmes difficiles \in NP

Problèmes faciles \in P

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

Exemple NP : watermelons



12 kg max

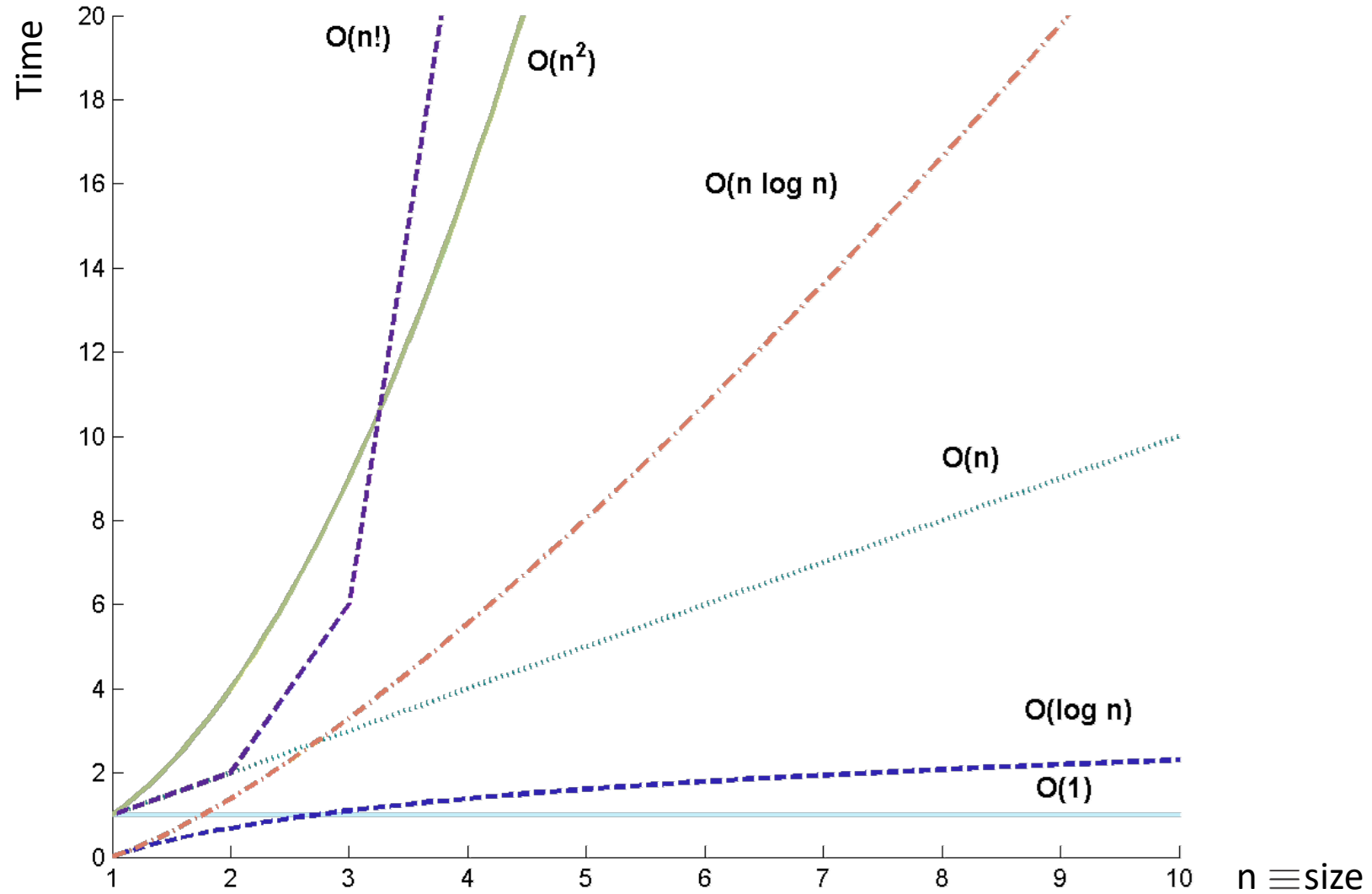
Solution possible



Algorithmique (2)

- En algorithmique, notation $\mathcal{O}()$
- Nommé *ordre* de la complexité. Obtenu si taille n du problème tend vers $+\infty$
- *Exemples :*
 $\mathcal{O}(1), \mathcal{O}(n), \mathcal{O}(n^2), \mathcal{O}(2^n), \mathcal{O}(n \log(n))$

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



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

Arranger selon un ordre

3. ALGORITHMES DE TRIS

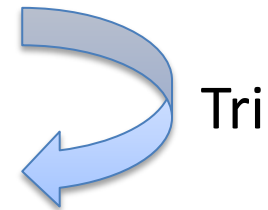
Algorithmes de tri

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.

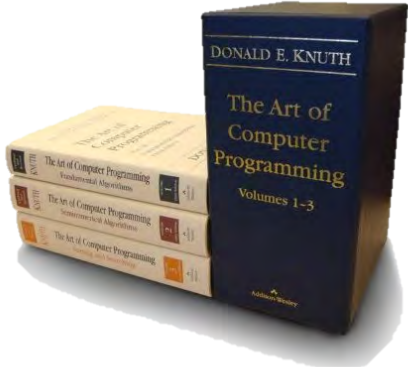
Méthode de tri

Autrement dit :

8	6	2	1	14	5	5	4
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
1	2	4	5	5	6	8	14
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]



Donald E. Knuth



L^AT_EX

A Bohemian in Exile

A REMINISCENCE

WHEN, many and extends dissolved a were found some have gone on to powerful chiefrain who,

$$(1.6) \quad \dot{g} = gg^{ij} \dot{g}_{i,j} = -2e^\psi Hg,$$

and thus, the volume of $M(t), |M(t)|$, evolves according to

$$(1.7) \quad \frac{d}{dt}|M(t)| = \int_{M(t_1)} \frac{d}{dt} \sqrt{g} = - \int_{M(t_1)} e^\psi H,$$

where we shall assume without loss of generality that $|M(t_1)|$ is finite, otherwise, we replace $M(t_1)$ by an arbitrary measurable subset of $M(t_1)$ with finite volume.

Now, let $T \in [t_1, T_+)$ be arbitrary and denote by $Q(t_1, T)$ the cylinder

$$(1.8) \quad Q(t_1, T) = \{ (x^0, x) : t_1 \leq x^0 \leq T \},$$

(a) At what speed does the proton enter the magnetic field? (3 pts)
 (b) Will the proton follow path a or path b? (1 pt)
 (c) What will the radius of this path be? (3 pts)
 (d) How long after it enters the magnetic field will the proton hit the back of the capacitor plate? (3 pts)

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

8	6	2	1	14	33	5	4
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]

Tri par sélection, autre représentation

	8
	5
	2
	6
	9
	3
	1
	4
	0
	7

Tri par insertion

// Insertion sort pseudo-code

Tant que il y a élément mal placé
on cherche sa bonne place
on le déplace à sa bonne place

- Le tri des cartes !



Code (pour référence)

```
// Insertion sort function
public static void sort(int[] a) {
    for (int i = 1; i < a.length; i++){
        int j = i;
        int tmp = a[i];

        while ((j > 0) && (a[j-1] > tmp)){
            a[j] = a[j-1];
            j--;
        }

        a[j] = tmp;
    }
}
```

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.

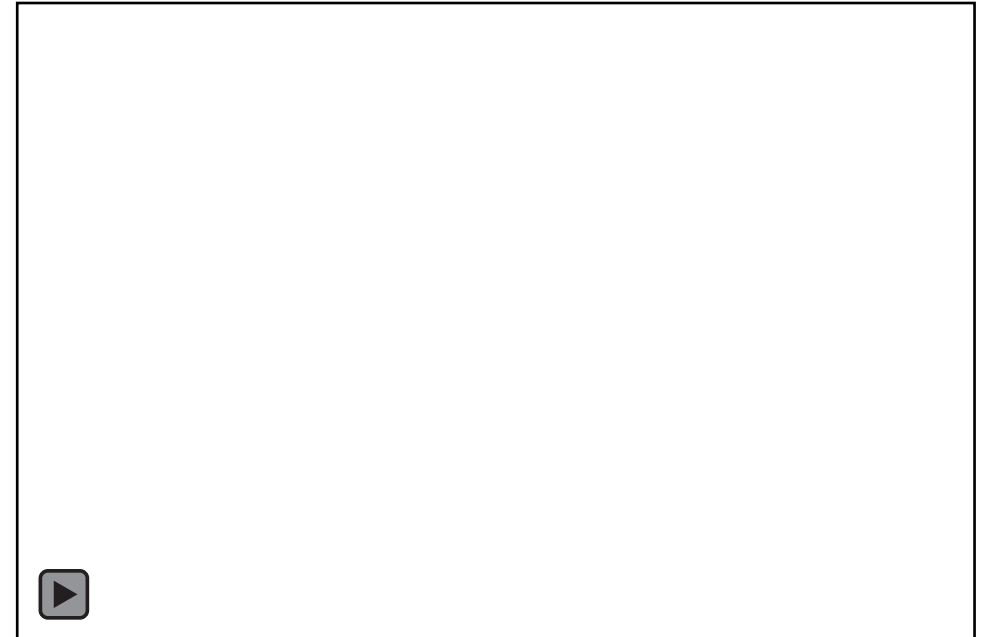
// Bubble sort pseudo-code

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]



Complexité des tris examinés

- Algorithmes vus sont les plus simples mais pas 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	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n^2)$
Mergesort	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n \log(n))$

Conclusion



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