



# Data Structure & Algorithms



Slide: Matthieu Jimenez  
Thème: Sébastien Mosser

Jimenez Matthieu  
TD #1, 21.09.2015

# Déroulement des TDs

---

- 8 SÉANCES DE 1H30
- 3 notes de Tds pour 40 % de la note finale



# Déroulement des TDs

---

- 21/09
- 28/09
- 12/10
- 26/10
- 9/11
- 23/11





Data Structure & Algorithms TD#1

Algorithme

**Petite Rappel ...**

Un  
Algorithm?

Un **algorithme** est une  
**méthode** pour **résoudre** un  
**problème.**

Code ou  
Pseudo-code ?

# Pseudo Code

---

Le **pseudo-code** est destiné à être **compris** par des **humains** et peut se rapprocher du code.



# Code

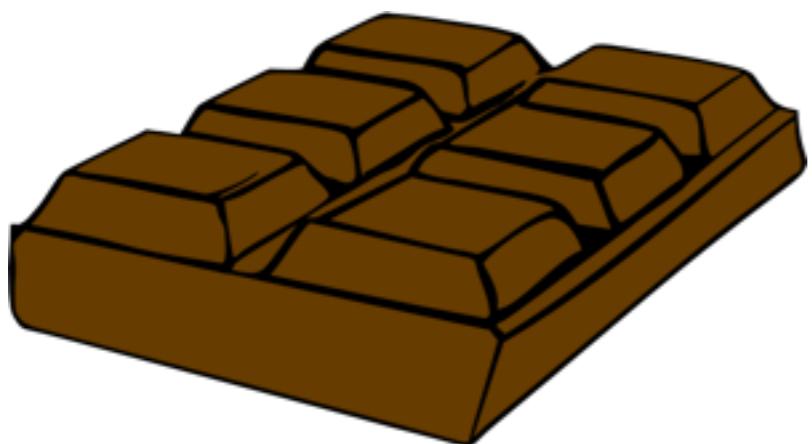
---

Là où le code est destiné à être interprété par la machine

```
        }
    endif; // twentyfifteen_setup
add_action( 'after_setup_theme', 'twentyfifteen_setup' );

/**
 * Register widget area.
 *
 * @since Twenty Fifteen 1.0
 *
 * @link https://codex.wordpress.org/Function_Reference/register_sidebar
 */
function twentyfifteen_widgets_init() {
    register_sidebar( array(
        'name'          => __( 'Widget Area', 'twentyfifteen' ),
        'id'            => 'sidebar-1',
        'description'   => __( 'Add widgets here to appear in your sidebar.', 'twentyfifteen' ),
        'before_widget' => '<aside id="%1$s" class="widget %2$s">',
        'after_widget'  => '</aside>',
        'before_title'  => '<h2 class="widget-title">',
        'after_title'   => '</h2>',
    ) );
}
```

# Exemple ?



Algorithme des chocolats

# Exemple de Pseudo-Code

---

input: liste des moyennes des élèves

output: chocolat

Pour toute les moyennes de la liste :

Si aucune note < 10:

l'enseignant de TD rapporte une boite de chocolat

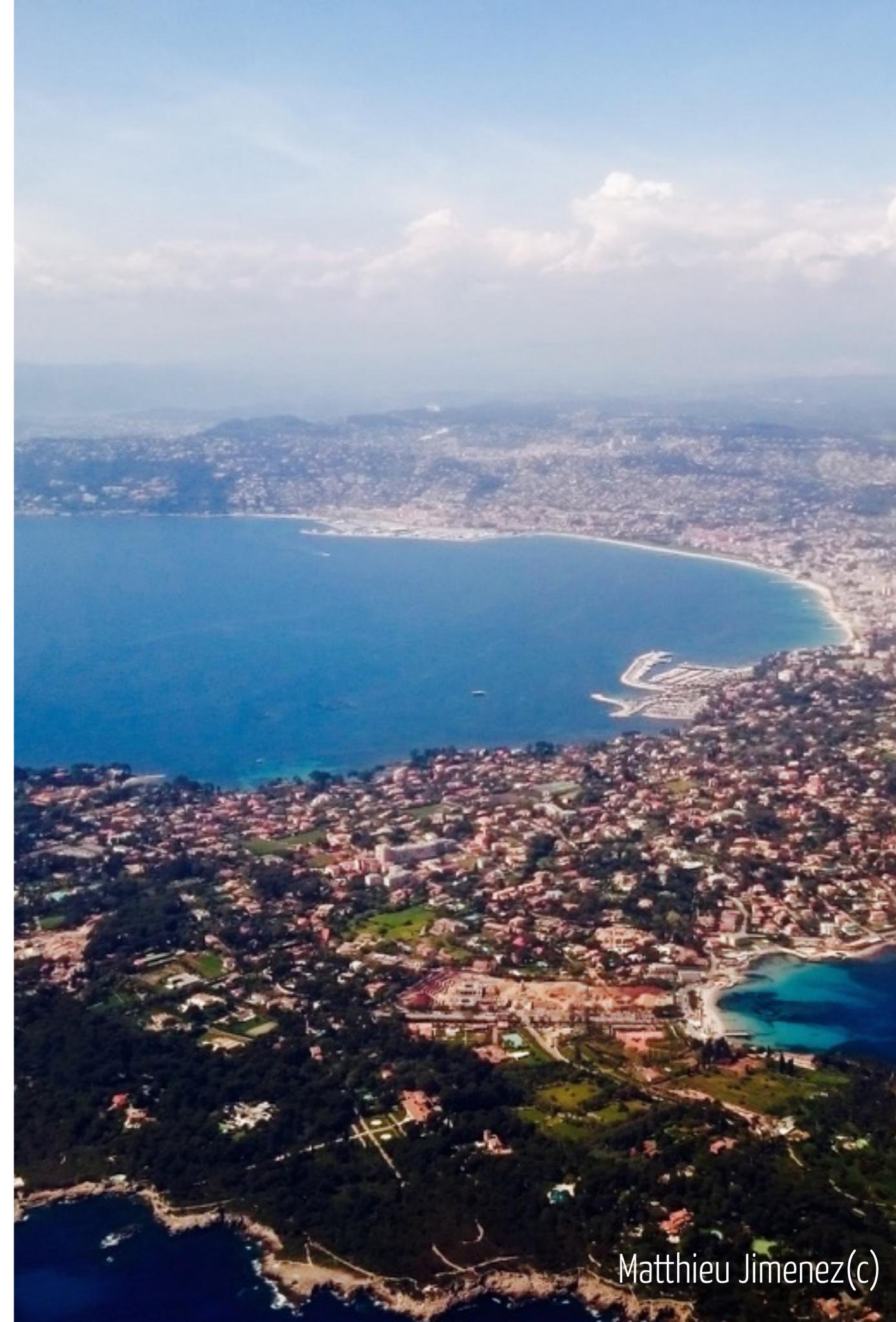
# Exemple de Code

---

```
public ChocolatBox algorithmeDesChocolats(List<Int> listEleve) throws  
NoChocolateException{  
  
    for(int i =0;i< list.size();i++){  
  
        if (list[i]<10)  
  
            throw new NoChocolateException();  
  
    }  
  
    return new ChocolatBox();  
}
```

# Exercice 1

---



Matthieu Jimenez(c)

# Exercice 1

---

Proposer un **algorithme** qui prend **en entrée une série de nombre** et qui **renvoie** le **premier plus petit élément** de cette série.

Exemple:

si la suite de nombre est 5,2,4,2,1,7,9,4,1,1

L'élément minimal **1** apparaît **3 fois** et sa première occurrence est **en 5 ème position**



Data Structure & Algorithms TD#1

Structure de données

# Structure de données ?

Une **Structure de**  
**données** permet de  
**stocker** des données ...

de manière logique  
afin de **simplifier** leur  
**traitement !**

# Structure de données existantes

---

- Une liste
- Un tableau
- Une hashmap
- Un graphe
- ...

# Quel utilité?

# Exemple Informatique

---

Je veux la **moyenne générale** de la classe:  
**2 façons** de gérer les notes:

**moyEleve1 = 12**

**moyEleve2 = 15**

**moyEleve3 = 14**

**moyEleve4 = 11**

**moyEleve5 = 10**

**List moyenneEleve=**

**[12,15,14,11,10]**

## Exercice 2

---



# Exercice 2

---

Proposer **deux pseudo-code** permettant de calculer la moyenne de la classe l'un **prendra en entrée la première façon** de gérer les données l'autre **la seconde.**

J'ai oublié de rentrer la  
moyenne de l'élève 6

!!!  
...

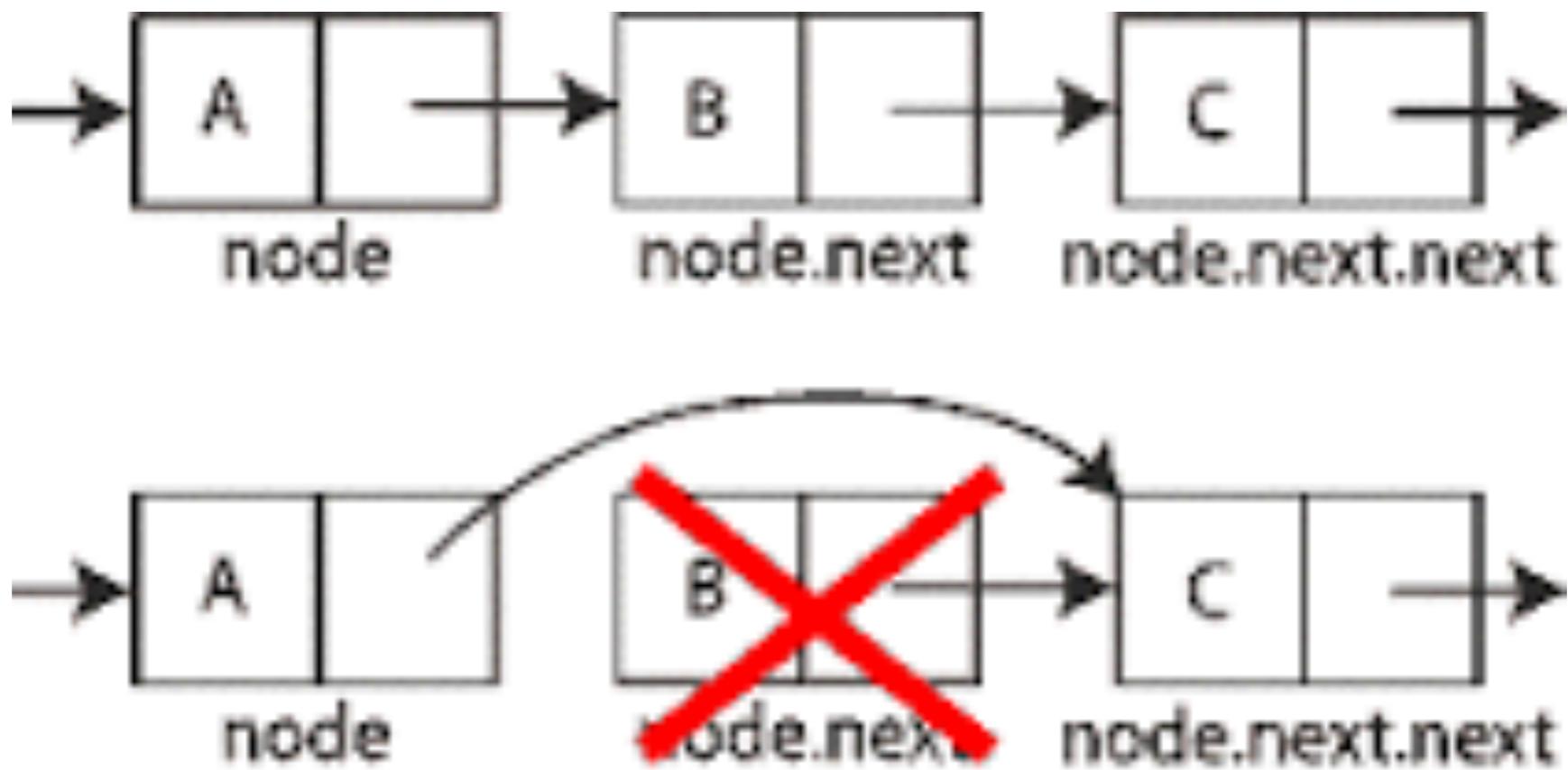
# Exercice 2b

---

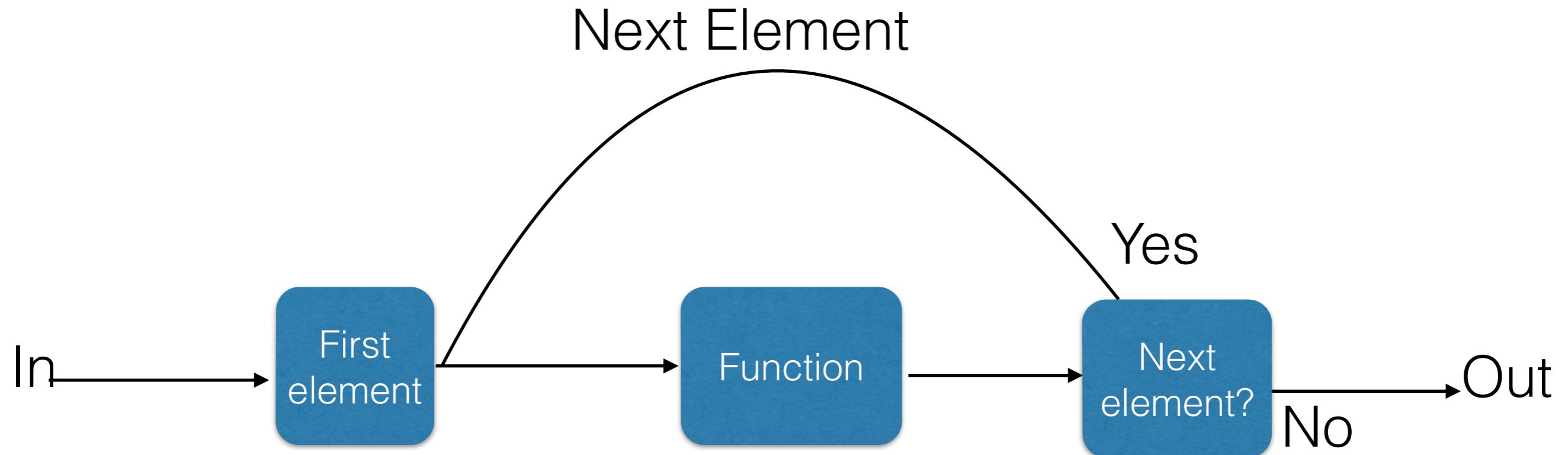
Que doit on faire pour **la prendre en compte** et **qu'est ce** que cela **change** à votre **pseudo code**?

**Votre conclusion?**

**Petite** Précision ...



Dans **cette première partie** de semestre  
Nous allons **essentiellement** travailler avec  
des **Listes et des Tableaux** ...



Que nous **pourrons explorer** à l'aide de  
**boucles (for, while)**

# Exercice 3

---





Les **mots de passe** de **compte utilisateurs** sont des données **très critiques** que l'on ne souhaite absolument **pas** voir **compromises**.



Si l'effort de **renforcer la sécurité** de ces mots de passe est à la **charge** des **entreprises**,



il convient également de **conseiller** les **utilisateurs** afin qu'ils ne choisissent pas des mots de passe **trop simple**.

cost-saving email services for business cost-saving email services for business cost-saving em

**AVSpamfilter.com**  
cost-saving email services for business

Add Email account

Email Account:  @watchingtime.com

Password:

Password (again):

Password strength: Weak Weak

Real name:

Spam Detection?

Passwords must be at least six characters and include three of the following four types: upper case letters, lower case letters, numbers and special characters.

---

[ [Email Accounts](#) | [Main Menu](#) | [Log Out](#) ]

[www.avspamfilter.com](http://www.avspamfilter.com)

Une **méthode** possible est d'obliger l'utilisateur à choisir un mot de passe comprenant **au minimum une majuscule, une minuscule et un chiffre.**



L'entreprise pour laquelle vous travaillez, a décidé d'appliquer **cette méthode**.



Elle vous a donc chargé de concevoir  
**l'algorithme** permettant cette **vérification**.

# Exercice 3

---

Proposez **un pseudo-code** reprenant le **style** de celui vu en cours:

input: Chaine de caractères

output: boolean

Note: Vous avez à votre disposition les fonctions que vous pouvez incorporer dans votre pseudo-code:

boolean isMinuscule(char c);

boolean isMajuscule(char c);

boolean isNumber(char c);



Data Structure & Algorithms TD#1

Complexité

Pourquoi faire **complexe**  
quand on peut faire **simple** ?

L'idée est de pouvoir évaluer **l'efficacité**  
d'un **algorithme**

« Pourquoi ne pas tester  
directement avec un  
ordinateur ? »

Réponse  
illustrée...

```
int minimum=0
for (int i=1; i<list.size();i++){
    if (list[i]<list[minimum]){
        minimum=i;
    }
}
return minimum;
```

```
for (int i =0; i<list.size();i++){
    bool k = true;
    for (int j=0; j<list.size();i++){
        if (list[i]>list[j]){
            k=false;
            break;
        }
    }
    if (k){
        return i;
    }
}
```



0,1s



0,1us



```
int minimum=0
for (int i=1; i<list.size();i++){
    if (list[i]<list[minimum]){
        minimum=i;
    }
}
return minimum;
```

```
for (int i =0; i<list.size();i++){
    bool k = true;
    for (int j=0; j<list.size();i++){
        if (list[i]>list[j]){
            k=false;
            break;
        }
    }
    if (k){
        return i;
    }
}
```



0,1s



0,1us

# OBJECTION!



Nintendo(c)

C'est la **puissance** de  
**l'ordinateur** qui a **joué** ici !

De même **différentes**  
**implémentations** d'un **même**  
**algorithme** peuvent donner de faux  
résultats ...

```
int minimum=0  
for (int i=1; i<list.size();i++){  
    if (list[i]<list[minimum]){  
        minimum=i;  
    }  
}  
return minimum;
```

```
for (int i =0; i<list.size();i++){  
    bool k = true;  
    for (int j=0; j<list.size();i++){  
        if (list[i]>list[j]){  
            k=false;  
            break;  
        }  
    }  
    if (k){  
        return i;  
    }  
}
```

$O(n)$

$O(n^2)$

```
int minimum=0  
for (int i=1; i<list.size();i++){  
    if (list[i]<list[minimum]){  
        minimum=i;  
    }  
}  
return minimum;
```



```
for (int i =0; i<list.size();i++){  
    bool k = true;  
    for (int j=0; j<list.size();i++){  
        if (list[i]>list[j]){  
            k=false;  
            break;  
        }  
    }  
    if (k){  
        return i;  
    }  
}
```



$O(n)$

$O(n^2)$

On va créer un **modèle** permettant **d'évaluer**  
le **temps d'execution** de **manière**  
**abstraite !**

# Condition au modèle

---

1. Chaque **instruction** de  
**pseudo-code** prend un  
**temps constant**



# Condition au modèle

---

2. Si une **ligne** est  
**exécutée plusieurs fois**  
(boucle), on **multiplie** la  
**constante** par le **nombre**  
**d'exécutions**



# Condition au modèle

---

3. Si une ligne appelle une fonction, on prend en compte le temps total utilisée par la fonction

```
void setup() {  
    Serial.begin(9600);  
  
    Serial.println("Before example function call.");  
    delay(1000);  
    example();  
    Serial.println("After example function call.");  
    delay(1000);  
}  
  
void loop() {  
}  
  
void example() {  
    Serial.println("During example function call.");  
    delay(1000);  
}  
When done, sketch returns to next instruction after function call.
```

Function call.

... sends sketch to function.

Function starts here.

Au final, la **complexité** sera exprimé en  
**fonction** de la **taille** de vos  
**éléments d'entrés !**

Mais,

si j'ai **des conditions** dans  
**mon algorithme?**

Pour répondre à cela, on va **considérer** la  
**complexité** dans **3 cas distincts**

# Best Case

---

Le **traitement** des données  
est **optimale**.

Par exemple:

trier une liste déjà trié  
(et encore... #stayTuned)



# Average Case

---

- On fait la **moyenne** de tout les **possibilités** de complexité



# WORST CASE

---

Le **traitement** des données  
est un **désastre**,  
on prend le pire chemin possible ...



**Attention** toutefois, on parle du  
**contenu** de la **structure de données** et **non**  
de sa **taille**

# Ne pas confondre avec

---

~~Best Case~~



~~Average Case~~



~~Worst Case~~



# Exercice 4

---



# Exercice 4a

---

Reprenez **votre algorithme** de **l'exercice 1** et calculer sa **complexité best case** et **worst case**.

Y a t il des différences entre vous?

# Exercice 4b

---

Reprenez **votre algorithme** de **l'exercice 3** et calculer sa **complexité best case** et **worst case**.

Y a t il des différences entre vous?



C'EST TOUT POUR AUJOURD'HUI