Information, Calcul et Communication (partie théorique)

Olivier Lévêque

EPFL, Sections de génie civil et science des matériaux

20 septembre 2019

Pourquoi un cours d'introduction à l'informatique et aux communications pour les étudiant-e-s en génie civil et en sciences des matériaux?

- L'informatique est devenue le 4e pilier de la culture dans notre société. (après la lecture, l'écriture et l'arithmétique)
- Elle constitue désormais une discipline scientifique à part entière (et bientôt également une discipline enseignée au gymnase): c'est la science du traitement automatique de l'information.
- L'informatique a non seulement changé notre société, mais aussi notre façon de faire de la science.
- De nos jours, tout-e ingénieur-e qui maîtrise les sciences du numérique a clairement un avantage sur les autres.

Structure du cours

Partie théorique Olivier Lévêque Vendredi 8h-11h	Partie pratique Jean-Philippe Pellet Vendredi 13h-16h
Algorithmes (6 semaines)	Programmation (7 semaines)
Semaine 7: test (30%)	Semaine 8: test (30%)
Applications (6 semaines)	Programmation (5 semaines)

Semaine 14: test final conjoint (40%)

Contenu détaillé du cours (partie théorique)

Semaine	Cours
1	Algorithmes: ingrédients de base
2	Complexité d'un algorithme
3	Récursivité
4	"Programmation" dynamique
5	Classes de complexité des problèmes
6	Problèmes d'optimisation discrète
8	Représentation binaire et échantillonnage
9	Reconstruction de signaux
10	Compression de données
11	Correction d'erreurs
12	Cryptographie et sécurité
13	Réseaux

Logistique du cours (partie théorique)

- Cours: Vendredi 8h15-10h en salle ~~BCH 2201~~ CM 1 3
- Exercices: Vendredi 10h15-11h15+, en salles:

  CM 1 120 (Lettres A à H), GC B3 31 (Lettres I à O)
  et GC C3 30 (Lettres P à Z) [attention temps de marche!]

- 1 enseignant, 1 assistant-doctorant et 8 assistants-étudiants sont à votre disposition: profitez-en!
- Egalement à votre disposition: transparents sur moodle.epfl.ch
- Livre du cours: "Découvrir le numérique", aux Editions PPUR , disponible au prix étudiant de 35.- à la libraire La Fontaine
- Et après ça? Utilisez le forum de discussion sur moodle!

Encore quelques conseils...
(valables pour d'autres cours également!)

- Venez (à l'heure) au cours et prenez des notes.
- N'hésitez pas à poser des questions, même en cours ☺
- Participez activement aux séances d'exercices.
- Retravaillez le cours et les exercices à la maison.
- Ne ratez pas le train...

Première partie: Algorithmes

Plan détaillé des prochaines semaines de cours:

Semaine	Cours
1	Algorithmes: ingrédients de base
2	Complexité d'un algorithme
3	Récursivité
4	"Programmation" dynamique
5	Classes de complexité des problèmes
6	Problèmes d'optimisation discrète

Illustration

$n = 5$

$\Rightarrow 5$

$\times 4$

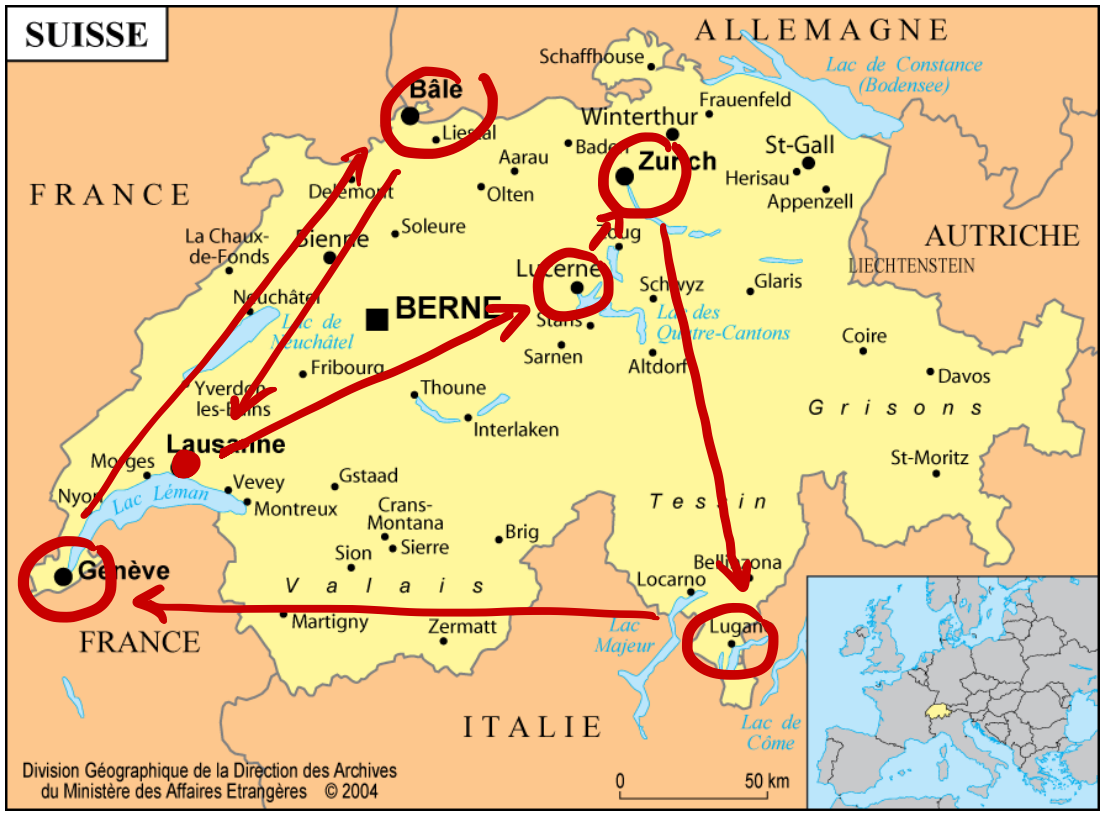
$\times 3$

$\times 2$

$\times 1$

$= 5!$

$= 120$ possibilités



$n = 26$

$\Rightarrow 26!$

$\simeq 4 \times 10^{26}$ possibilités

Qu'est-ce qu'un algorithme?

Algorithme

Un algorithme n'est pas un programme.
Un algorithme est la description des étapes simples menant à la résolution d'un problème; c'est la partie conceptuelle d'un programme.

Programme

Un programme est l'implémentation d'un algorithme dans un langage donné et sur un système particulier.

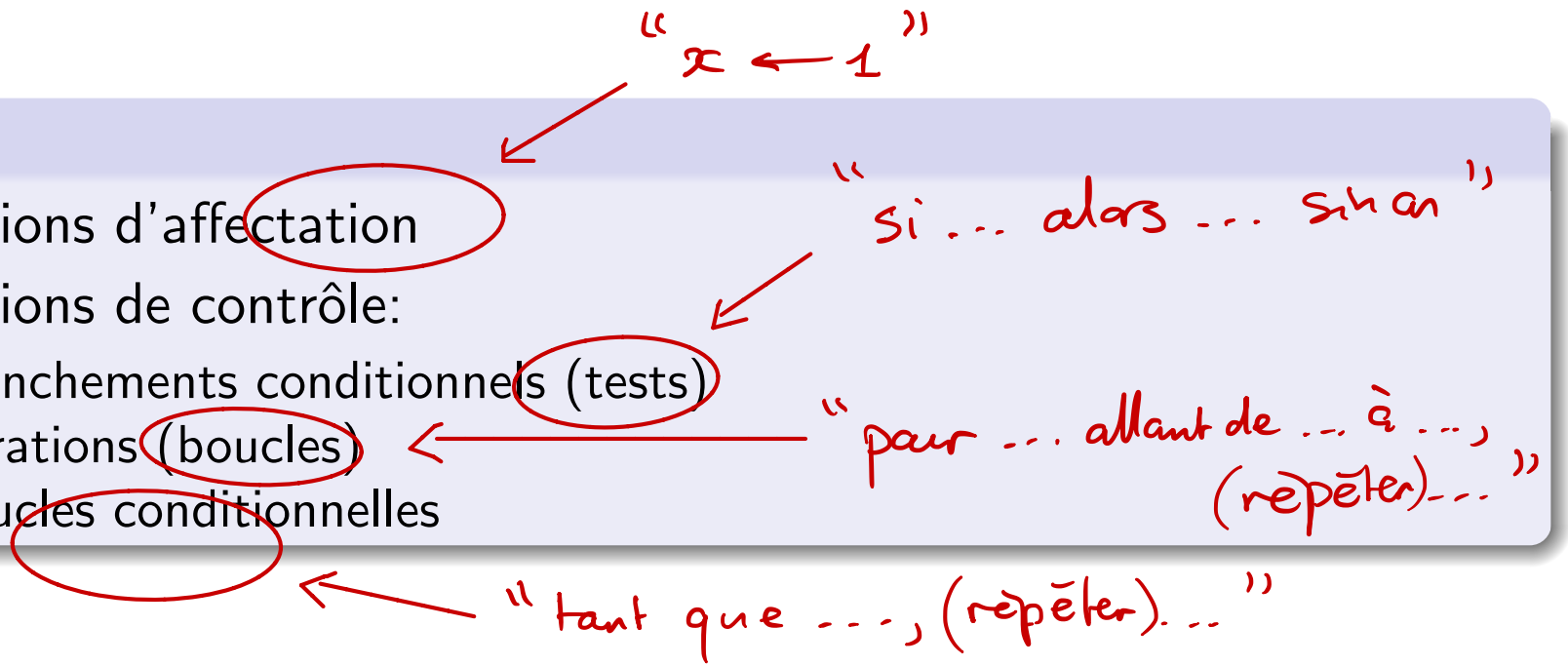
Ingrédients de base des algorithmes

Données

- données d'entrée
- données de sortie
- variables internes

Instructions

- instructions d'affectation
- instructions de contrôle:
  - branchements conditionnels (tests)
  - itérations (boucles)
  - boucles conditionnelles



Voyons des illustrations de ces divers éléments dans les pages qui suivent...

Recherche du minimum dans une liste

Exemple : Trouver la personne la plus jeune parmi une liste de 8 personnes : $L = (7, 18, 82, 75, 43, 5, 25, 12)$

Ici, le minimum vaut 5.

Remarques préliminaires :

- Si la liste est petite, le problème paraît trivial. Mais imaginez maintenant que la taille de la liste $n = 10^6$...
- Pour simplifier la notation, on considère une liste de nombres dans ce qui suit, mais l'algorithme ci-dessous s'applique également à toute liste d'éléments qui peuvent être ordonnés (dates de naissance, mots du dictionnaire, ...)

Description informelle d'un algorithme qui résout ce problème :
(il y a d'autres possibilités)

Recherche du minimum dans une liste

1. assigner à une variable x la valeur du premier élément de la liste
2. a-t-on atteint la fin de la liste ?
   si oui, sortir x
3. passer à l'élément suivant dans la liste
4. est-ce que cet élément est plus petit que x ?
   si oui, remplacer x par la valeur de cet élément
5. recommencer au point 2

Description formelle (pseudo-code)

Recherche du minimum dans une liste

```
entrée: L liste de nombres, n taille de L
sortie: x nombre le plus petit de la liste L
i ← 1
x ← L(1)
Tant que i<n :
    i ← i+1
    Si L(i) < x, alors :
        x ← L(i)
Sortir : x
```

Version alternative du pseudo-code

Recherche du minimum dans une liste

entrée : L liste de nombres entiers, n taille de la liste

sortie : x nombre le plus petit de la liste L

$x \leftarrow L(1)$

Pour i allant de 2 à n :

    Si $L(i) < x$, alors :

        $x \leftarrow L(i)$

Sortir : x

(Remarquer que le "tant que" est en quelque sorte inutile dans la version précédente ; dans tous les cas, on doit parcourir la liste en entier.)

Remarques :

- Le pseudo-code n'est pas un langage informatique à proprement dit, mais juste une manière de formaliser les idées ; ceci entraîne bien sûr une certaine souplesse dans les notations (nous reviendrons sur celles-ci tout au long du cours).
- L'entrée n (taille de la liste L) n'est pas forcément nécessaire ; celle-ci pourrait être calculée au début de l'algorithme.
- Il se peut que la liste ait deux éléments (ou même plus) ayant la même plus petite valeur, mais ceci n'affecte pas la sortie de l'algorithme.

Tous différents ?

Parmi une liste de 3 éléments, identifier si ceux-ci sont tous différents les uns des autres.

Algorithme

entrée : $L = (L(1), L(2), L(3))$ liste de 3 éléments

sortie : variable binaire s (oui/non)

$s \leftarrow$ oui ← affectation d'une variable interne

Si $L(1) = L(2)$, alors : $s \leftarrow$ non

Si $L(1) = L(3)$, alors : $s \leftarrow$ non

Si $L(2) = L(3)$, alors : $s \leftarrow$ non

} 3 tests successifs

Sortir : s ← sortie de l'algorithme

Remarque :

On aurait pu écrire de manière équivalente :

> Si $L(1)=L(2)$ ou $L(1)=L(3)$ ou $L(2)=L(3)$, alors : sortir non
>
> sinon : sortir oui

mais certainement pas oublier une de ces 3 conditions, ni écrire :

> $S \leftarrow$ non
>
> Si $L(1)=L(2)$, alors : $S \leftarrow$ oui
>
> si $L(1)=L(3)$, alors : $S \leftarrow$ oui
>
> si $L(2)=L(3)$, alors : $S \leftarrow$ oui
>
> sortir : S

Tous différents ? (bis)

Parmi une liste de n éléments, identifier si ceux-ci sont tous différents les uns des autres.

Algorithme

entrée : L liste de n éléments, n taille de la liste

Sortie : variable binaire s (oui/non)

$s \leftarrow$ oui

Pour i allant de 1 à n-1 :

  Pour j allant de i+1 à n :

    Si $L(i) = L(j)$, alors : $s \leftarrow$ non

Sortir : s

deux boucles imbriquées (énumération de toutes les paires (i, j))

Conjecture de Syracuse

Pour finir, mentionnons un algorithme intéressant utilisant une boucle conditionnelle "Tant que" : cet algorithme décrit une suite mathématique de nombres entiers positifs, partant d'un certain nombre x, donné en entrée de l'algorithme.

Cet algorithme a beau être parfaitement bien défini, à l'heure actuelle, on ne sait toujours pas s'il s'arrête pour toute valeur d'entrée x (même si on pense que oui, car on l'a vérifié pour tous les $x \leq 1.25 \cdot 2^{62}$ ...)

Suite de Syracuse

Entrée : x nombre entier positif

Sortie : L liste de nombres entiers positifs

$L \leftarrow (x)$

Tant que $x \neq 1$:

- Si x est pair, alors :
  - $x \leftarrow x/2$
- Sinon :
  - $x \leftarrow 3x+1$
- $L \leftarrow L + (x)$ ("ajouter x à la liste L")

Sortir : L

[6]:
```
# Suite de Syracuse
from matplotlib.pyplot import *
x=int(input("Position de départ: "))
L=[x]
while (x!=1):
    if x%2==0:
        x=x/2
    else:
        x=3*x+1
    L=L+[x]
xlabel("temps de vol")
ylabel("altitude")
plot(L)
print("Temps de vol: ",len(L))
print("Altitude maximale: ", int(max(L)))
```

```
Position de départ: 1001
Temps de vol:  143
Altitude maximale:  21688
```