Introduction à la programmation

Gilles Trombettoni

IUT MPL-Sète, département info

Septembre 2019

Qu'est-ce qu'un algorithme ?

Algorithme

- Méthode détaillée (séquence d'instructions) permettant de résoudre un problème (une classe de problèmes).

  Un algorithme est clairement spécifié, dans un langage formel, sans ambiguïté. Il est donc exécutable sur un ordinateur.

- En pratique, on désigne souvent par algorithmes les parties difficiles d'un logiciel.

Origine du mot *algorithme*

Nom d'un mathématicien perse du 9^e siècle :
Abu Abdullah Muhammad Ibn Musa **al-Khwarizmi**

Exemples d'algorithmes

- plus court chemin d'une ville à une autre, d'un appart à un autre
- algorithme de chiffrement (cryptographie) ou de stéganographie (ex : cacher un message dans une image)
- algorithme de séquençage d'ADN : déterminer l'ordre d'enchaînement des nucléotides (A, T, G, C) ou les gènes d'un fragment d'ADN donné
- algorithme d'aide au diagnostic médical
- ordonnancement de tâches (ex : d'un chantier)
- traitement automatique de la langue naturelle : traduction, construction d'un réseau lexical (cf. `www.jeuxdemots.org`)
- reconnaissance d'une plante à partir d'une photographie (cf. `plantnet.org`)
- algorithme de guidage (d'une fusée, d'un missile, etc.)
- algorithme d'optimisation continue ou de résolution d'un système d'équations
- algorithme de décision collective (politique)

Jeux de mots : construction d'un réseau lexical

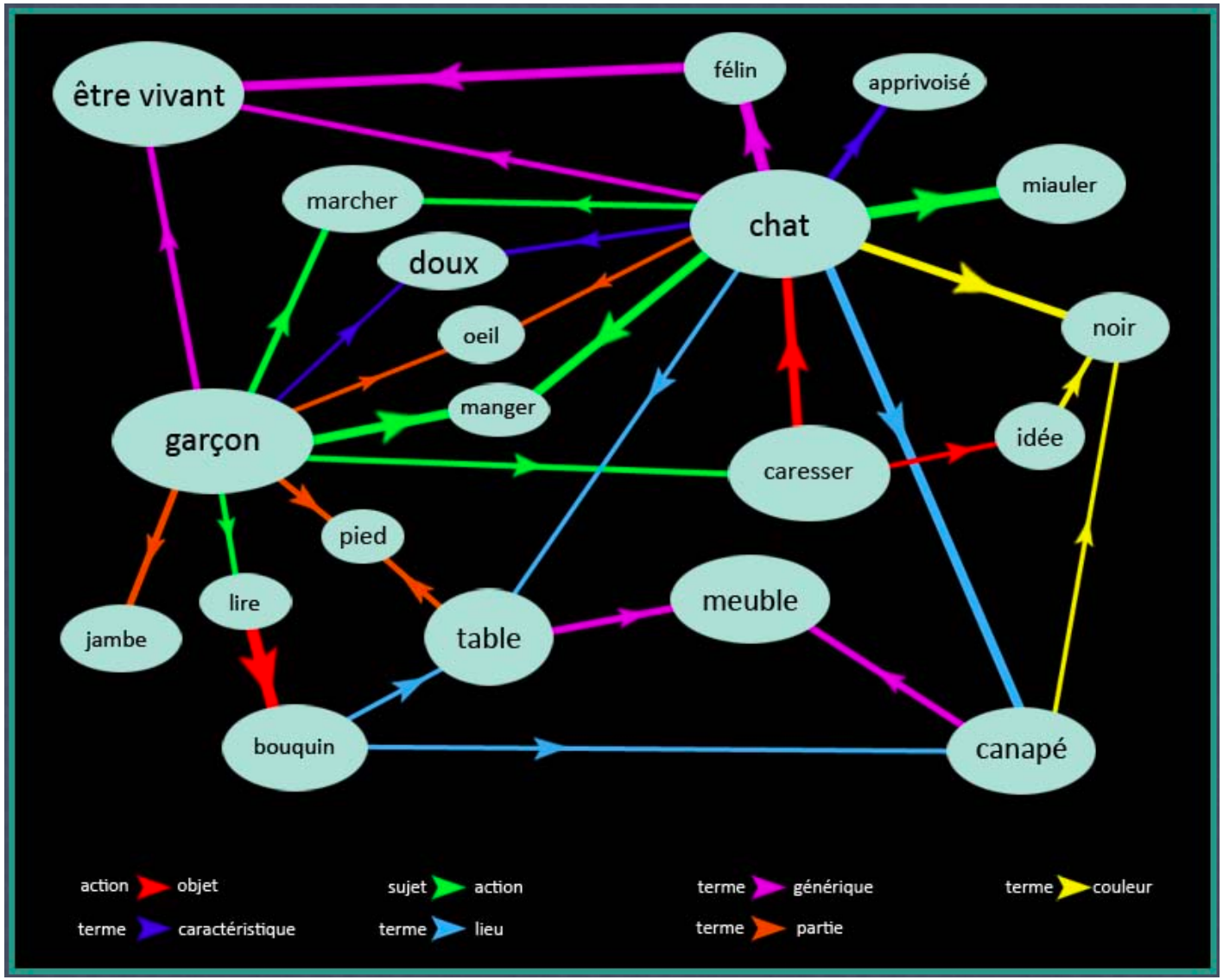

Photo extraite du cours de M. Alain Joubert et `www.jeuxdemots.org`

Pl@ntNet : identification des plantes

Pl@ntNet : plateforme participative pour l'identification des plantes et la collecte de données botaniques (cf. `plantnet.org`)

Optimisation globale

But : trouver (par un algorithme travaillant avec des **intervalles**) les valeurs des variables qui minimisent une fonction tout en respectant des contraintes.

```
Variables
  x2, x3, x4, x5 in [1e-7,0.5];  x6 in [0,0.901];
  x7 in [0,0.274];  x8 in [0,0.69];  x9 in [0,0.998];

Minimize
   x2*(log(x2) - log(x2 + x4)) + x4*(log(x4) - log(x2 + x4))
   + x3*(log(x3) - log(x3 + x5)) + x5*(log(x5) - log(x3 + x5))
   + 0.92*x2*x8 + 0.746*x4*x6 + 0.92*x3*x9 + 0.746*x5*x7;

Subject to  x6*(x2 + 0.159*x4) - x2 = 0;
            x7*(x3 + 0.159*x5) - x3 = 0;
            x8*(0.308*x2 + x4) - x4 = 0;
            x9*(0.308*x3 + x5) - x5 = 0;
            x2 + x3 = 0.5;  x4 + x5 = 0.5;
```

Commençons par des algorithmes simples

Les algos précédents sont d'un niveau avancé (recherche), pas d'un premier cycle. Nous devons commencer par des algos plus simples. Exemple : détermination du nombre le plus petit dans une série.

```
Algo nombreMinimum
 // Action : Détermination et affichage de la plus petite valeur
 //          d'une série de 100 nombres saisis au clavier.
 // Stratégie : stockage du plus petit nombre rencontré jusqu'à
 //             présent dans une variable (petit)
Variables
    nb, petit, i : entier
Début
    afficher("Donner un premier nombre") ; saisir(petit)
    i <-- 1
    TantQue i < 100 faire
       afficher("Donner un nombre") ; saisir(nb)
       Si nb < petit Alors
          petit <-- nb
       FinSi
       i <-- i + 1
    FinTantQue
    afficher("Le plus petit nombre de la série est : ", petit)
Fin nombreMinimum
```

De l'algorithme au programme

Les micro-processeurs ne comprennent pas ces algorithmes. Ils ne comprennent pas le langage algorithmique qui n'est pas assez détaillé. Plusieurs étapes sont nécessaires pour qu'un ordinateur fasse tourner un algorithme. Voici le schéma le plus simple :

1. Les programmeurs traduisent l'algorithme dans un langage de programmation (Ada, C, Java, Python, etc.) : l'algorithme devient un programme.
2. **Compilation :**
   Programme source $\longrightarrow$ **programme exécutable**
   L'exécutable est écrit en **langage machine** (fait de 0 et de 1) et est compréhensible par le processeur.
3. **Exécution :** Le programme exécutable est exécuté par le processeur d'un ordinateur.

Voici des exemples de programmes en C et en Ada.

Exemple de programme C (fichier `petit.c`)

```
#include<stdio.h>
int main () {
int petit, nb, i;
        printf( "Saisir␣un␣premier␣nombre␣:␣" );
        scanf("%d", &petit);
        i = 1;
        while (i < 100) {
            printf("Saisir␣un␣nombre␣:␣" );
            scanf("%d", &nb);
            if (nb < petit) {
                petit = nb;
             }
            i++;
        }
        printf("Le␣plus␣petit␣nombre␣de␣la␣serie
            ␣est␣:␣%d␣\n", petit);
}
```

Exemple de programme Ada (fichier `Petit.adb`)

```
with text_io; use text_io; ...

Procedure nombreMinimum is
Begin
    put("Donner un premier nombre ") ;
    get(petit);
    i := 1 ;
    While i < 100 loop
        put("Donner un nombre : ") ;
        get(nb);
        if nb < petit then
            petit := nb;
        end if;
        i := i + 1;
    end loop;
    put("Le plus petit nb de la serie est :");
    put(petit);
End nombreMinimum;
```

Compilation versus interprétation

Compilation d'un programme C

1. Compilation : `gcc petit.c -o petit`
   (`petit` est un programme en langage machine)
2. Exécution : `./petit`

Interprétation

- Un programme (appelé parfois machine virtuelle) exécute les instructions du programme source.
- Exemple (en Python) : `python petit.py`
- Remarque : le programme `python` prend en compte l'ordinateur sur lequel il travaille.

Programme en Python (fichier `petit.py`)

```
petit=int(input("Saisir␣1er␣nombre␣:␣"))
i=1
while (i < 100):
    nb=int(input("Saisir␣un␣nombre␣:␣"))
    if (nb<petit):
        petit=nb
    i += 1
print("Le␣plus␣petit␣nombre␣de␣la␣serie␣est␣:␣"
   + repr(petit) + "\n")
```

Compilation en langage Java

Java est un langage ni compilé, ni interprété :

1. La commande `javac` compile le programme source dans un langage « intermédiaire » (entre le langage source et le langage machine). C'est un langage connu par les développeurs de Java... et par la commande `java`.

   Exemple : `javac Petit.java` génère le fichier `Petit.class`

2. La commande `java` interprète les instructions du fichier en langage intermédiaire.

   Exemple : `java Petit[.class]`

Dans les versions récentes, la machine virtuelle `java` (appelée JVM - *Java Virtual Machine*) effectue une **compilation à la volée**...
Voici le programme `Petit.java`.

```
import java.util.Scanner;
public class petit {
  public static void main (String [] argv) {
      int petit, nb, i;
      Scanner scanner = new Scanner(System.in);
      System.out.print("Saisir␣1er␣nombre␣:␣");
      petit = scanner.nextInt();
      i = 1;
      while (i < 5) {
        System.out.print("Saisir␣un␣nombre␣:␣");
        nb = scanner.nextInt();
        if (nb < petit) {
            petit = nb;
        }
        i++;
      }
      System.out.println("Le␣plus␣petit␣nombre␣
         de␣la␣serie␣est␣:␣" + petit);
  }
}
```

Vie d'un logiciel

Plusieurs étapes seront détaillés dans les enseignements de « Analyse, conception et développement d'applications » (ACDA) :

1. Analyse des besoins, spécifications
2. Ecriture des algorithmes
3. Développement des programmes
4. Test des programmes

Modules d'algo-prog

Modules d'algorithmique et de programmation

Semestre	Nom	Numéro	Durée	Contenu
1	prog1	M1102	49h	Bases prog
1	prog1-bis	M1103	42h	Structures de données
2	prog2-obj	M2102	60h	Prog à objets
3	prog3-avance	M3103	50h	Prog récursive, complexité
4 (2 groupes)	prog4-fonc.		15h	Prog fonctionnelle (Scheme)

Programmation au semestre 1

- Prog1 (avant Toussaint) : Instructions de base, variables, tableaux, sous-programmes (procédures et fonctions), algorithmes de base, traduction dans un langage de programmation (Ada, Java ou Python), [enregistrements].
- Prog1-bis (après Toussaint) : Bases de programmation à objets (classes/instances), références, structures de données.
- Rythme : 1h de cours + 3 $\times$ 2 h de TD ou TP machine.

Langage utilisé

En TD : langage « algorithmique » (pseudo-code, en français).

En TP (salles machines) : « vrai » langage de programmation, selon l'enseignant :

- Ada : assez proche du langage algorithmique proposé, mais peu utilisé dans l'industrie ; ou
- Java : langage très utilisé, mais plus difficile d'accès ; ou
- Python : langage en croissance et utilisé au lycée

Evaluation

Note basée sur « petits » contrôles et/ou rendus de TP et contrôle de fin de demi-semestre.

- *noteTD* : moyenne des « petits » contrôles et des rendus de TP éventuels
- *notePartiel* : contrôle de fin de demi-semestre
- Note de prog1 $=$ 40% *noteTD* $+$ 60% *notePartiel*