L'algorithme

pourquoi et comment le définir et l'enseigner

Emmanuel Beffara

Laboratoire d'Informatique de Grenoble

Inspé de Grenoble

École thématique DEMIMES, 5 avril 2022

Introduction

De quoi on va parler

Algorithme

- Comment l’enseigne-t-on ?

De quoi on va parler

Algorithme

- Comment l'enseigne-t-on ?
- Comment la notion interagit-elle avec les autres enseignements ?

De quoi on va parler

Algorithme

- Comment l'enseigne-t-on ?
- Comment la notion interagit-elle avec les autres enseignements ?
- De quelle discipline dépend-elle ?

De quoi on va parler

Algorithme

- Comment l’enseigne-t-on ?
- Comment la notion interagit-elle avec les autres enseignements ?
- De quelle discipline dépend-elle ?
- Qu’est-ce que c’est, au juste ?

De quoi on va parler

Algorithme

- Comment l’enseigne-t-on ?
- Comment la notion interagit-elle avec les autres enseignements ?
- De quelle discipline dépend-elle ?
- Qu’est-ce que c’est, au juste ?
- Et pourquoi veut-on l’enseigner ?

Le plan de la suite

L’algorithme comme objet scientifique

- Élaboration de la notion d’algorithme
- Différentes façons de le définir

L’algorithme comme objet d’enseignement

- Ce qu’on enseigne dans différents contextes
- Quelques éléments didactiques

L’algorithme, du savoir-faire à l’objet d’étude

Procédures codifiées

L’algorithme comme savoir-faire :

- méthodes de calcul posé liées à chaque système de numération
- approximation de racines carrées chez les babyloniens
- calcul de diviseurs communs chez Euclide
- résolution d’équations chez Al-Khwarizmi

Caractéristiques fondamentales de ces procédures :

- communicables, sans ambiguïté
- sans contrainte *a priori* autre que la compréhension de l’interlocuteur

Machines programmables

Transposer la procédure dans le matériel, coder l'information dans un support physique :

- machines à calculer :
  - Pascaline (1645)
  - arithmomètre (1820)
- machines programmables :
  - métiers à tisser (Bouchon 1725, Jacquard 1801)
  - boîtes à musique (1796)
- machines à calculer programmables :
  - machine analytique de Babbage (1834)
  - machines à calculer universelles (Zuse 1941, ENIAC 1946)

Ce qui se passe :

- préciser la procédure jusqu'à la rendre *mécanique*
- l'exprimer dans un langage formalisé et contraint

Calculabilité

Interrogations sur les limites des procédures systématiques :

- les grands problèmes de l'Antiquité
- la résolution d'équations
- la logique formelle et le problème de la décision (XXe siècle)

Pour établir que quelque chose ne peut pas exister, il faut lui donner une définition formelle :

- approches algébriques : nombres rationnels, algébriques, transcendants ; théories des groupes et des corps…
- approches opératoires : modèles de calcul (on y reviendra)

Au delà du calcul

L'informatique s'est développée comme science et comme technologie, au delà de la programmation de procédures de calcul séquentielles :

- parallélisme et calcul distribué
- entrées et sorties
  (capteurs, actionneurs, interfaces homme-machine)
- communication en réseau
- calcul quantique et autres

L'algorithmique s'est enrichie en conséquence.

Des enjeux actuels en recherche

Conception d'algorithmes :

- analyse de données pour la décision, la planification
- recherche opérationnelle, optimisation
- calcul numérique, simulation
- cryptologie et sécurité
- …

Analyse d'algorithmes et de programmes :

- modélisation et vérification
- démonstration automatique ou assistée
- …

Fondements :

- modèles de calcul
- complexité
- correspondance preuve programme
- …

Sondage

Exercice

Au sujet de ces différentes propositions données dans la suite, on pose plusieurs questions.

1. Lesquelles de ces propositions sont des algorithmes ? Pourquoi ?
2. Lesquelles de ces propositions sont des programmes ? Pourquoi ?
3. Grouper les propositions qui sont le même algorithme.

1. Retrancher le plus grand nombre au plus petit jusqu'à ce que l'un des deux soit nul. Le PGCD est celui qui n'est pas nul.

2. Notons $a \wedge b$ le PGCD de a et b. Pour tous a et b, on a $a \wedge b = a \wedge (b - a)$. Par ailleurs, pour tout a on a $a \wedge 0 = a$.

3. Entrées : deux entiers a et b
   Tant que $b \neq 0$,
   poser la division euclidienne $a = bq + r$,
   $b \leftarrow a$,
   $a \leftarrow r$.
   Répondre a.

4. Entrées : deux entiers a et b
   Tant que $b \neq 0$,
   si $a > b$
   alors $a \leftarrow a - b$
   sinon échanger a et b.
   Répondre a.

6.

```
def pgcd(a, b):
    while b != 0:
        a, b = a % b, a
    return a
```

7.

```
pgcd a 0 = a
pgcd a b = pgcd (a % b) a
```

8.

```
int pgcd(int a, int b) {
    while (b > 0)
        if (a > b) { a -= b; }
        else { a ^= b; b ^= a; a ^= b; }
    return a;
}
```

Comment définir l’algorithme

Différentes conceptions

Qu’est-ce qu’un algorithme ?

- informaticien : c’est ce qu’on va programmer pour résoudre un problème
- mathématicien : c’est juste une méthode de calcul
- technophile : c’est des réseaux de neurones pour faire rouler les voitures autonomes
- citoyen inquiet : c’est une façon de nous espionner et de nous influencer

Définitions non formelles

Wikipédia (le 17 mars 2022) :

> *Un algorithme est une suite finie et non ambiguë d'instructions et d'opérations permettant de résoudre une classe de problèmes.*

Version plus élaborée (Modeste 2012) :

> *Un algorithme est une procédure de résolution de problème, s'appliquant à une famille d'instances du problème et produisant, en un nombre fini d'étapes constructives, effectives, non ambigües et organisées, la réponse au problème pour toute instance de cette famille.*

Ce genre de définition a ses limites :

- pas assez précis pour établir des résultats de (non) calculabilité
- permet l'analyse d'algorithmes mais pas forcément les résultats généraux sur les problèmes algorithmiques

Une référence : Knuth 1969

Dans *The Art of Computer Programming*, pas de définition mais des critères qui s'appliquent à un ensemble fini de règles :

- finitude (critère d'arrêt)
- définition précise (sans ambiguïté)
- entrées (prises avant l'exécution dans un ensemble spécifié)
- sorties (qui ont une relation spécifiée avec les entrées)
- effectivité (opérations réalisables à la main)

Une machine idéalisée sert de référence pour les résultats formels.

Une référence : Cormen et al 2004

Dans *Introduction à l'algorithmique* de Cormen, Leiserson, Rivest et Stein (traduit en français par Cazin et Kocher) :

> *Voici une définition informelle du terme algorithme : procédure de calcul bien définie qui prend en entrée une valeur, ou un ensemble de valeurs, et qui donne en sortie une valeur, ou un ensemble de valeurs. Un algorithme est donc une séquence d'étapes de calcul qui transforment l'entrée en sortie.*

Cette définition ne fait pas référence à un problème à résoudre !

Définitions formelles

Machine de Turing :

> *Une machine de Turing est un sextuplet* $(\Sigma, \Gamma, Q, i, f, \delta)$ *où* Γ *est un ensemble fini non vide (l'alphabet de travail),* Σ *est un sous-ensemble non vide de* Γ *(l'alphabet d'entrée et de sortie) contenant un élément distingué* $\varnothing$ *(le symbole vide) non élément de* Σ*,* Q *est un ensemble non vide (les états),* i *et* f *sont des éléments de* Q *(l'état initial et l'état final) et* δ *est une fonction de* $Q \times \Gamma$ *dans* $Q \times \Gamma \times \{-1, 0, 1\}$ *(la table de transition).*

Et je vous épargne la suite de la définition.

- beaucoup de détails sans importance
- un choix de modélisation particulier
- une inadéquation avec la notion intuitive

Définitions formelles

D’autres choix :

- le λ-calcul, les fonctions récursives
- le modèle de Von Neuman
- les automates cellulaires
- tous les langages de programmation

Deux constats fondamentaux :

- aucune définition formelle ne peut être satisfaisante
- toutes définissent la même notion de calculabilité (thèse de Church-Turing)

Si tous les modèles sont équivalents, il y a certainement une notion sous-jacente.

Programmes ou algorithmes

Version 1 :

```
def pgcd(a, b):
    while b != 0:
        a, b = a % b, a
    return a
```

Version 2 :

```
pgcd a 0 = a
pgcd a b = pgcd (a % b) a
```

Ces deux *programmes* calculent la même chose de la même façon : ils représentent le même *algorithme*.

Mais les ensembles d'entrée et de sortie ne sont pas spécifiés ?

L’usage médiatique

On est gouvernés par les algorithmes !

Référence à quelques systèmes informatiques particuliers :

- aide à la décision
- captation et de monétisation de l’attention

Confusion entre « algorithme » et « intelligence artificielle »

- focalisation sur l’apprentissage et les données massives
- ne rend justice ni à l’algorithmique ni à l’IA

La préoccupation est légitime

- influence réelle et vécue d’un objet mystérieux et fantasmé
- le jargon cache des partis pris idéologiques (prétendue objectivité de la machine, solutionnisme technologique)

Des questions à débattre

L'algorithme existe-t-il sans spécification ?

- Le problème est la raison d'être de l'algorithme.
- Qu'est-ce qu'on étudie quand on fait une preuve de correction ?

Un algorithme serait une *classe d'équivalence de programmes* ?

- Comment définir cette équivalence ?
- Le programme existe indépendamment du problème, alors où émerge la notion de problème ?

Et quelques autres sujets de réflexion :

- La place du langage, naturel et artificiel.
- P vs NP comme symptôme de la difficulté de définition.
- Une recette de cuisine est-elle un algorithme ?

L’algorithme comme objet d’enseignement

Pourquoi enseigner l'algorithme ?

Différents buts possibles, parmi lesquels :

- démystifier l'informatique et ce qu'elle produit
- contribuer à la culture générale scientifique
- développer la "pensée informatique" comme une façon parmi d'autre d'aborder des problèmes
- équiper les élèves de vocabulaire et d'outils pour formuler ou étudier des méthodes de calcul
- donner des occasions de raisonner et faire de la logique
- apprendre à de futurs informaticiens à concevoir leurs programmes
- poser les bases pour comprendre des résultats de calculabilité et de complexité

Quelles préconceptions ?

(selon Modeste et Rafalska 2018)

Au lycée :

- Procédure effective
- Liée à la résolution de problème pour 2/3 des gens
- Critère d'efficacité qui émerge

En L1 :

- Référence systématique à la notion de problème
- Critère d'efficacité dans les filières d'informatique

En L2 d'informatique :

- Importance majeure de la notion de preuve

Dans l’enseignement supérieur

Suivant l’approche de Knuth ou Cormen :

- pas vraiment de définition
- la programmation comme justification et référence
  (les algorithmes sont fait pour être programmés)
- résultats formels (correction, complexité) basés sur l’intuition
  partagée après plusieurs années de programmation

Les modèles de calcul apparaissent en master (ou L3 pour les plus audacieux) pour aborder la calculabilité et la complexité des problèmes.

Dans l’enseignement secondaire

- Quelques manuels tentent des définitions plutôt de l’ordre de l’intuition ou emploient des analogies (recette de cuisine)
- Les “vrais” algorithmes sont souvent considérés comme des brouillons de programmes
- La notion se construit en même temps que la pratique de la programmation dans un langage et un environnement spécifiques (Scratch au collège)
- Focalisation sur les structures perçues comme fondamentales :
  - structures de contrôle
  - variables
  - structures de données
- Un peu de preuve

Dans l’enseignement primaire

- Développement du savoir-faire calculatoire
  - mise en avant des suites d’opérations
  - références à la géométrie :
    déplacements, construction de figures
- Verbalisation des procédures
- Participe à l’apprentissage de la démarche scientifique
- Les sujets proprement informatiques sont anecdotiques

En médiation

L’informatique débranchée :

- Utilisation d’objets concrets, tangibles
- Séparer les questions et problèmes fondamentaux des dispositifs techniques où ils vivent pour réduire le “bruit”
- Alterner entre le rôle de la machine (pour construire l’intuition des problèmes et de leur résolution sur le ressenti individuel) et le rôle du programmeur (pour raisonner et verbaliser)

La médiation en informatique ne se limite pas à cela, bien sûr.

Éléments didactiques

De faire à faire faire

(d’après Samurçay et Rouchier 1985)

Élaborer un algorithme consiste à rendre communicable une procédure complète pour accomplir une tâche donnée :

- préciser l’ensemble des opérations et leur agencement
- éliminer l’implicite pour rendre chaque étape claire

Le texte doit être entièrement produit avant l’exécution :

- pas de communication avec l’opérateur
- dissociation stricte entre la planification et l’exécution

Analyse d’algorithmes

Pour prouver la correction, il faut :

- préciser l’attendu, au moyen d’assertions précises ;
- interpréter l’algorithme tel qu’il est écrit,
  pas tel qu’on voudrait qu’il agisse.

La procédure prend pleinement le statut d’objet :

- dissociation de la procédure suivie et du problème étudié
- nécessité d’une étude exhaustive des entrées possibles

```python
def dichotomie(liste, x):
    # Préconditions: x est un entier, liste est une
    # liste d'entiers triée dans l'ordre croissant
    n = len(liste)
    if n == 0:
        return False
    g, d = 0, n - 1
    while d - g > 0:
        m = (g + d) // 2
        if liste[m] >= x:
            d = m
        else:
            g = m
    return liste[g] == x
```

L’état et les variables

L’introduction de la *variable* est un difficulté classique :

- la notion d’affectation est radicalement nouvelle
- en conflit avec le calcul littéral, usages différents

Plus généralement, c’est la notion d’*état* qui est fondamentale :

- l’algorithme prescrit un transformation d’un système
- chaque instruction définit un changement d’état du système
- une variable est un morceau de l’état que l’on crée explicitement

Une variété d’usages qu’il est utile de distinguer :

- constante (paramètre exogène ou endogène)
- compteur
- accumulateur
- temporaire
- indicateur

Les compétences en jeu

(d’après Declercq)

Évaluer : attribuer mentalement une valeur à un programme

Anticiper : en posture de programmeur, décrire l’enchaînement des opérations avant le début de l’exécution

Décomposer : transformer un problème complexe en un ensemble de problèmes plus simples équivalent

Généraliser : inférer un problème général à partir d’une instance, repérer la répétition de traitements ou de données suivant un même schéma

Abstraire : ignorer des informations non pertinentes et créer des solutions où la façon de résoudre un problème peut être “abstraite” à l’aide d’une interface pertinente

La question de la preuve

L'étude des algorithmes comporte toujours une dimension logique.

- Spécification des problèmes
  nécessite la formulation d'énoncés précis
- Description de l'effet des instructions sur l'état
- Identification des invariants
  un analogue du raisonnement par récurrence,
  nécessité de généralisation
- Peut-on voir la démonstration mathématique comme une activité algorithmique ?

Discutons