Les algorithmes

Jean-Luc Chabert,

Le concept est surtout aujourd'hui l'apanage des informaticiens, qui s'en nourissent pour composer les programmes d'ordinateurs. Au cours de sa longue histoire, il a connu plusieurs significations, qui avaient toutefois en commun la référence à la notion de règles opératoires.

Où rencontre-t-on des algorithmes ?

Bien que le mot paraisse technique, la vie de tous les jours nous donne à voir de nombreux algorithmes. Une recette de cuisine, une règle de grammaire en sont des exemples. Une fiche-tricot ressemble aussi de façon frappante à la description d'un algorithme. Voici, par exemple, celle du point des « côtes piquées » [1].

Nombre de mailles pour la symétrie : multiple de 3 + 2, + 1 maille lisière à chaque extrémité.

1er et 3e rang : 1 maille lisière ; *2 mailles à l'endroit, 1 maille à l'envers* ; répéter de * à * ; terminer par 3 mailles à l'endroit, 1 maille lisière.

2e rang : tricoter les 3 mailles comme elles se présentent.

4e rang : tout à l'endroit.

Répéter toujours ces 4 rangs.

On a, dans tous les cas, affaire à l'énumération d'une liste ordonnée de règles et de consignes à respecter en vue de parvenir à un résultat. Une condition essentielle pour que la procédure considérée soit à proprement parler un algorithme est que le nombre d'étapes à effectuer soit fini. Cette condition de bon sens est évidemment vérifiée dans le cas de notre fiche- tricot : même si son rédacteur suggère de « répéter toujours », il va de soi que l'algorithme a vocation à prendre fin lorsque assez de mailles ont été tricotées. Cependant, pour des algorithmes plus compliqués dont le déroulement varie en fonction des données de départ, il peut être difficile de s'assurer que cette condition de finitude est bien vérifiée.

Est-ce différent en mathématiques ?

En dehors du tricot, de la cuisine et de la grammaire, les algorithmes sont très présents en mathématiques, et même aussi anciens que la discipline elle-même. Un exemple particulièrement emblématique d'algorithme mathématique est une procédure de calcul du plus grand commun diviseur (PGCD) entre deux nombres, appelée algorithme d'Euclide. Pour calculer le PGCD de deux nombres a et b, on effectue la division eucli- dienne du plus grand (disons a) par le plus petit, c'est-à-dire la division en entiers avec reste (sans virgule), ce qui donne a = bq + r, où q est le quotient et r le reste (compris entre 0 et b ? 1). Si r = 0, alors c'est que a est divisible par b et donc que le PGCD de a et b est b lui-même. Dans le cas contraire, on remarque que tout diviseur commun à a et b est aussi diviseur commun à b et à r, et réciproquement, de sorte que le PGCD de a et b est aussi celui de b et r. On effectue alors la division euclidienne de b par r, et on recommence, encore et encore. Chaque fois, les nombres en jeu sont de plus en plus petits. Le processus finit donc par s'arrêter, en produisant un couple de nombres dont l'un est un diviseur de l'autre, et est donc le PGCD des deux, et donc aussi (en raisonnant de proche en proche) le PGCD de a et b. L'algorithme d'Euclide vérifie bien les conditions demandées aux algorithmes, notamment la finitude. Il comporte aussi l'idée d'un processus itératif, avec cette succession de divisions euclidiennes, comme dans l'exemple de la fiche tricot.

Quand la notion a-t-elle été explicitée ?

L'appellation d'« algorithme d'Euclide » est un brin anachronique, à la fois parce qu'Euclide lui-même, vers 300 av. J.-C., n'utilisait pas ce terme et parce que le point de vue du géomètre grec était un peu différent du nôtre. L'origine du mot est vraisemblablement arabe. Il s'agirait de la déformation d'al-Khwarizmi, nom de l'auteur du premier traité d'algèbre, au IXe siècle. Lorsqu'au XIIe siècle les ouvrages arabes ont commencé à être traduits en latin, le système de numération décimale que nous connaissons s'est diffusé en Europe. Très adapté pour effectuer les opérations arithmétiques, il dut toutefois batailler ferme pour s'imposer face à l'abaque, version occidentale du boulier chinois. Le terme algoritmus est alors introduit pour désigner les procédés de calcul utilisant l'écri- ture décimale, par opposition à ceux utilisant des jetons sur un abaque. Malgré la souplesse et la simplicité des calculs à l'aide du système de numération décimale, la victoire définitive des algoristes sur les abacistes est tardive : c'est seulement à la Révolution française que l'usage de l'abaque est interdit à l'école et dans l'administration.

Au XVIIIe siècle, l'Encyclopédie de d'Alembert définit le terme comme « l'arithmétique par chiffres », ainsi que « la pratique de l'algèbre » et, plus généralement, « la méthode et la notation de toute espèce de calcul ». En 1666, déjà, Leibniz s'intéressait à une « langue caractéristique universelle » dans laquelle les raisonnements mathématiques seraient de simples calculs. L'idée pointe donc d'une mécanisation par le calcul pour résoudre des problèmes. Au XIXe siècle, des logiciens tels que Charles Babbage, George Boole, Gottlob Frege ou Giuseppe Peano tentent de théoriser le raisonnement mathématique en « algébrisant » la logique. Avant l'avènement de l'informatique, le terme d'algorithme désigne alors tout procédé de calcul rendu systématique, voire automatique, par l'indication de règles précises. Au XXe siècle, enfin, sous l'impulsion de la science informatique, la condition de finitude acquiert une place essentielle, et la question est clairement posée de savoir ce que l'on peut faire ou non avec des algorithmes.

Y a-t-il un rapport entre logarithme et algorithme ?

Ce ne sont que des anagrammes. Au début du XVIIe siècle, l'Écossais Napier invente la notion de logarithme qui fournit un procédé opératoire efficace pour effectuer rapidement des multiplications en les transformant en additions. Le mot est construit sur une racine grecque. En quelque sorte, la fonction logarithme intervient dans un algorithme de calcul des multiplications. Celui-ci, à la base du fonctionnement des règles à calcul, n'a été supplanté que dans les années quatre-vingt par l'apparition des calculatrices de poche.

Existe-t-il toujours un algorithme permettant de répondre à une question donnée ?

En 1900, le mathématicien David Hilbert propose une liste de questions mathématiques non résolues comptant parmi les plus importantes de l'époque. En particulier, existe-t-il une méthode (un algorithme) permettant de savoir si une équation diophantienne* donnée a ou non des solutions ? Si l'on avait trouvé un processus général répondant positivement à ce problème, les choses auraient été simples, mais ce ne fut pas le cas. Or, pour pouvoir affirmer que la réponse au problème de Hilbert est négative, c'est-à-dire affirmer que l'on ne pourra jamais exhiber un algorithme, parce qu'il n'en existe pas, on est contraint de formaliser de façon très précise les notions qui gravitent autour de celle d'algorithme.

Les logiciens se sont attelés à une telle formalisation dans les années trente. Entre 1931 et 1936, Kurt Gödel, Alonzo Church et Stephen Kleene introduisent des notions de fonctions récursives, formalisant la notion de fonction effectivement calculable.

Cette appellation fait référence aux fonctions dont on peut calculer la valeur à l'aide d'un algorithme pour toutes valeurs fixées des variables. Les fonctions récursives les plus simples sont l'addition et la multiplication, dont les écoliers apprennent dès leur plus jeune âge les algorithmes de calcul. Les fonctions récursives, concept mathématique formalisé, traduisent-elles correctement la notion floue et intuitive d'algori- thme ? La « thèse » de Church affirme que oui. C'est sur la base de cette affirmation, d'ordre métamathématique et qui ne peut pas être prouvée, que Matijasevic a répondu négativement, en 1970, au problème de Hilbert sur les équations diophantiennes.

Quel lien y a-t-il avec les programmes d'ordinateur ?

En 1922, Hilbert (encore lui) pose de manière générale le problème de la décision : existe-t-il une procédure générale permettant en un nombre fini d'étapes de dire si un énoncé mathématique est vrai ? Cette question donne l'occasion à l'Anglais Alan Turing d'introduire, en 1936, la notion de « machine de Turing » (qui lui permet de répondre négativement à la question de Hilbert). Il s'agit d'une machine abstraite, à la fois très simple dans son fonctionnement et capable d'effectuer, de façon plus ou moins rapide, toutes les opérations que l'on peut demander à une machine, donc à un ordinateur [fig. 1].

Grosso modo, cette machine est composée de trois parties : un ruban, suite infinie de cases dans lesquelles on écrit des lettres ou des symboles tirés d'un alphabet fini ; une unité centrale, qui ne prend qu'un nombre fini d'états ; et une tête de lecture/écriture reliant l'unité centrale et le ruban. Turing montre qu'il y a équivalence entre la calculabilité effective au sens de Church (c'est-à-dire à l'aide des fonctions récursives) et la calculabilité par sa machine. Ainsi, le fonctionnement de cette machine formalise la notion intuitive d'un algorithme de calcul. Dans nos ordinateurs, descendants de la machine de Turing, le programme n'est donc pas autre chose qu'un algorithme (pour peu qu'il ne tourne pas en boucle) et un langage informatique est une langue permettant d'écrire un algorithme que notre ordinateur sait lire et effectuer.

Comment comparer des algorithmes ?

L'importance croissante des algorithmes a conduit à la naissance d'un nouveau domaine scientifique : l'algorithmique ou science des algorithmes. L'algorithmique n'a pas pour but de produire des algorithmes, mais d'étudier des questions relatives aux algorithmes d'un point de vue général : par exemple, comme on l'a vu, la possibilité de l'existence d'algorithmes pour résoudre certains types de problèmes, ou encore la preuve d'algorithmes, c'est-à-dire la preuve de ce que les algorithmes considérés se terminent bien en un temps fini et répondent bien à la question posée.

Les preuves d'algorithmes sont une arme, encore à l'état d'ébauche, qui permettra peut-être un jour de réduire de façon drastique les bugs informatiques, ces erreurs dans la programmation qui font qu'un algorithme censé faire une chose en fait en réalité une autre sans qu'on le sache. On peut aussi vouloir comparer deux algorithmes différents répondant à une même question, pour savoir quel est le plus économique en temps ou en place. L'évaluation se fait en étudiant d'une part le nombre d'opérations qu'ils requièrent, d'autre part la place nécessaire pour garder en mémoire les données intermédiaires utiles pour l'avancement des calculs. On parle de complexités respectivement temporelle et spatiale. Par exemple, pour ce qui est du tricot, la complexité temporelle pourrait être le nombre total de mouvements à effectuer, tandis que la complexité spatiale serait le nombre d'aiguilles nécessaires pour mener le travail à bien. Selon les données initiales, le temps mis par un algorithme et la place qu'il utilise peuvent se révéler variables, ce qui fait que, quand on change ces données, on change aussi l'ordre des algorithmes en terme d'efficacité. Pour faire un classement objectif, on tient alors compte en général de la complexité « au pire », c'est-à-dire, pour chaque algori-thme, du temps et de l'espace nécessaires au traitement des données qui se révèlent pour lui les plus « encombrantes », à savoir les plus coûteuses en temps et en espace. Plutôt qu'une complexité au pire, on peut aussi classer les algorithmes par une complexité moyenne sur toutes les données initiales possibles. C'est ainsi que la complexité est susceptible de multiples variantes, d'où la nécessité d'être toujours précis dans la formulation des énoncés.

Peut-on produire du hasard à partir d'algorithmes ?

La façon la plus simple, sinon la plus naturelle, d'obtenir du hasard est d'effectuer de véritables expériences aléatoires, par exemple une succession de tirages à pile ou face. Il ne s'agit pas alors de simulation, et on peut établir des tables de nombres au hasard, ou nombres aléatoires, en utilisant de tels tirages. Mais pour ne pas toujours réutiliser les mêmes tables, il faudrait en établir de grandes quantités. Avec l'avènement des ordinateurs, on a été amené, pour des besoins pratiques, à utiliser des générateurs de nombres aléatoires. Ce que l'on obtient ainsi, ce ne sont pas des nombres aléatoires, puisqu'ils sont produits à l'aide d'un algorithme, donc à partir d'un processus purement déterministe. Toutefois, sous certaines conditions, certains algorithmes (comme certains phénomènes physiques ou chimiques) fournissent des successions de sorties qui, bien que déterminées, n'en présentent pas moins une apparence erratique. Des nombres obtenus avec de tels algorithmes sont dits pseudo-aléatoires ; en pratique, leur succession offre l'apparence du hasard.

Qu'est-ce qu'un algorithme probabiliste ?

Comme un algorithme classique, un algorithme probabiliste donne une réponse au bout d'un nombre fini d'étapes. La différence : la réponse fournie n'est pas toujours exacte. Elle est donnée avec une certaine incertitude que l'on pré-cise en terme de probabilité. C'est en particulier le cas du « test de primalité de Miller-Rabin ».

Celui-ci, comme tout test de primalité, a pour fonction de déterminer si un entier N donné est un nombre premier ou non. Les méthodes classiques étant trop longues à mettre en oeuvre pour de grandes valeurs de N, l'idée est d'accepter de perdre un peu de fiabilité au profit d'une plus grande rapidité. Le test de Miller-Rabin sur N s'effectue à l'aide d'un entier auxiliaire a choisi au hasard : si N et a ne vérifient pas certaines relations, alors N n'est pas premier. Dans le cas contraire, on ne peut rien dire de façon certaine. Mais si, en répétant le test avec d'autres valeurs de a, on obtient encore et toujours le même résultat, alors la probabilité que N ne soit pas premier devient de plus en plus faible. L'algorithme de Miller-Rabin fonctionne donc en effectuant des essais sur N avec diverses valeurs de a, jusqu'à ce que l'une d'entre elles permette d'établir que N n'est pas premier ou que la probabi- lité que N soit premier soit grande. En pratique, si un nombre N passe le test de Miller-Rabin pour une vingtaine de valeurs de a sans qu'aucune d'elles ne per- mette de conclure que N n'est pas premier, alors N est considéré comme presque sûrement premier.

Les algorithmes ont-ils une influence sur les mathématiques contemporaines ?

En mathématiques s'est toujours posée la question de l'existence de certains objets. Par exemple, existe-t-il des nombres transcendants, c'est-à-dire des nombres ne vérifiant aucune équation algébrique à coefficients entiers ? Il y a deux façons de répondre à cette question. La première est dite constructive, elle consiste à exhiber un nombre transcendant, par exemple le nombre *. La seconde façon de répondre à la question consiste à démontrer, comme l'a fait Georg Cantor à la fin du XIXe siècle, la nécessaire existence de nombres transcendants sans en exhiber aucun. Ainsi, la preuve de l'existence théorique d'objets mathématiques a longtemps été considérée comme suffisante en soi, bien que le début du XXe siècle ait vu éclore l'opinion contraire : l'école dite intuition- niste, initiée par Brouwer, rejette les preuves reposant sur le « principe du tiers exclu », affirmant que tout énoncé mathématique est, soit vrai, soit faux. Dans ce type de preuves, pour démontrer l'existence d'un objet, on commence par montrer que sa non-existence conduit à une absurdité : le principe du tiers exclu est alors invoqué pour affirmer l'existence de l'objet concerné.

L'ordinateur, quant à lui, ne peut pas se satisfaire de résultats non constructifs, c'est-à-dire qui n'exhibent pas effecti- vement l'objet envisagé. Jusqu'à récemment, l'arithmétique fournissait des bataillons de théorèmes dont l'énoncé commençait par : « Il existe un entier n0 tel que, dès que l'entier n est supérieur à n0 , alors... » Un moyen d'obtenir un énoncé affranchi de cette condition sur n est de mener une vérification exhaustive sur tous les n plus petits que n0, en se servant de l'ordinateur. Mais, bien sûr, une telle vérification n'est envisageable que si l'on connaît une valeur explicite de n0, ce que seule une preuve constructive permet d'obtenir. Ainsi, la possibilité ouverte par l'informatique d'obtenir des théorèmes valables pour tous les entiers, et pas seulement pour ceux plus grands qu'une certaine valeur, a incité les chercheurs à s'intéresser aux valeurs explicites. Quand celles-ci se révèlent trop grandes pour que les ordinateurs actuels soient en mesure de traiter les cas résiduels, il convient de montrer qu'on peut remplacer ces valeurs par des valeurs plus petites ; c'est aujourd'hui l'objet d'un nombre non négligeable de publications.

Jean-Luc Chabert,