Les algorithmes peuvent-ils tout résoudre ?

Jacques Dubucs

L’œil du philosophe

Les algorithmes peuvent-ils tout résoudre ?

Jacques Dubucs

Un algorithme est une recette conduisant à son but sans qu'on n'ait à faire preuve d'inventivité. Il en existe pour allumer les cigarettes (en l'absence de vent, frotter une allumette et l'approcher de l'extrémité libre du cylindre de papier tandis qu'on aspire par l'autre extrémité). Il n'y en a pas pour savoir s'il conviendrait, en France, de repenser l'action publique (on a une vague idée que ce ne serait pas une mauvaise chose, mais comment s'y prendre pour en être tout à fait certain ?). Ces deux exemples suggèrent qu'il y a des algorithmes là où les questions sont précisément posées (une cigarette est allumée ou non). Et qu'il n'y en a pas là où les questions sont vagues et indéterminées (quand faut-il « repenser l’action publique » et qu'est-ce au juste que repenser ?). Mais en vérité, les choses ne sont pas ainsi. C'est ce que les lignes qui suivent vont s'efforcer d'expliquer

On fait parfois remonter les algorithmes à l'algébriste persan Al-Khwârismî, mort à Bagdad vers 850, et dont le nom « algorithme » dérive. Bien entendu, il n'en est rien. Les Grecs avaient déjà, sous un autre nom, des algorithmes parfaitement définis. L'un des plus connus est le « crible d'Eratosthène », du nom du philosophe et mathématicien qui, au III° siècle avant notre ère, dirigeait la bibliothèque d'Alexandrie. Cet algorithme, encore en usage aujourd'hui sous des formes plus raffinées, sert à déterminer si un nombre entier *n* est premier ou non. Il consiste à essayer de diviser *n* par les entiers successifs *k* pris à partir de 2, jusqu'à ce que l'une des deux situations se réalise : soit on a trouvé une division qui « tombe juste ». *k* est alors un diviseur de *n*, qui n'est donc pas premier, et on s'arrête là. Soit, à force d'incrémenter *k, o*n est arrivé à une valeur de *k* égale à *n*. Dans ce cas, on s'arrête aussi, avec cette conclusion : puisqu'il est impossible que la division d'un entier par un autre plus grand que lui « tombe juste », c'est que *n* est premier.

L'important, c'est que l'opération, dans tous les cas, s'arrête. Que le verdict soit négatif (l'entier n'est pas premier), ou positif (il l'est), elle conduit dans tous les cas à une *décision* en un nombre fini d'étapes. Les notions comme « être un nombre pair » ou « être un nombre premier » sont ce qu’on appelle des notions *décidables* : on peut leur associer des algorithmes conduisant à statuer sur le fait qu’un objet pertinent les satisfait ou non. Bien entendu, ces notions ne se limitent pas au règne des mathématiques : « être une carte Navigo valide » est un concept décidable, et l'algorithme qui, appliqué à des rectangles de plastique, délivre un verdict positif (le portillon

s'ouvre) ou négatif (il reste bloqué) est exécuté des milliers de fois chaque matin dans les transports parisiens.

Les choses commencent à devenir intéressantes lorsqu'on se demande, un problème étant donné, s'il existe un algorithme pour le résoudre ou bien, un concept étant proposé, s'il est ou non décidable. On constate alors une remarquable asymétrie entre les réponses OUI et les réponses NON. De quoi s'agit-il ? Prenez une question de capacité, caractéristique de cette asymétrie : êtes-vous capable de soulever 100 kg ? Peut-on construire, à la règle et au compas, un carré de surface équivalente à celle d'un cercle donné ? Peut-on construire, toujours à la règle et au compas, un cube de volume double d'un cube donné (c'est le « problème délien », posé par les sophistes au VIe siècle, dans lequel il s'agissait de rendre à Apollon un hommage renouvelé en lui construisant un autel de forme cubique deux fois plus volumineux que l'ancien) ? Dans la réponse à ces trois questions, l'asymétrie est la règle : si vous le faites, c'est que vous pouviez le faire, mais, si vous ne le faites pas, ceci ne signifie pas que vous ne pourrez jamais y arriver. Peut-être encore un effort, encore un essai, et vous parviendrez à soulever les haltères que pour l'heure vous n'arrivez pas à arracher du sol, ou à trouver une nouvelle construction géométrique qui va résoudre le problème pour le moment resté insoluble.

L'idée selon laquelle une réponse négative pourrait être apportée à des questions de ce type germe très tard, vers la fin du XVIIIème siècle, et elle est probablement une caractéristique de la science moderne. A cette époque, les gens commencent à se dire que des tentatives avortées pendant 2 000 ans s'expliquent peut-être par la vanité de leur objectif. En 1775, l'Académie Royale des Sciences décide ainsi qu'elle n'ouvrira plus les lettres de quadrateurs ou d'inventeurs de machines à mouvement perpétuel, au motif d' « *une longue expérience, qui a suffi pour [la] convaincre du peu d'utilité qui résulterait pour les Sciences de l'examen de toutes ces prétendues solutions* ».

Toutefois, une chose est d'être convaincu qu'un problème n'aura pas de solution, une autre est de démontrer qu'on a raison de l'être. Nous devons la première de ces *démonstrations* au mathématicien français Evariste Galois. Celui-ci établit qu'à partir du cinquième degré, il est inutile de chercher à résoudre des équations algébriques « par radicaux », c'est-à-dire au moyen d'une nombre fini d'opérations élémentaires sur ses coefficients. Inutile, donc, de chercher à procéder comme on le fait pour le second degré, avec la recette bien connue pour trouver les racines de l'équation $a + bx + cx^2 = 0$: en calculer le discriminant $\Delta = b^2 - 4ac$ puis, s'il est positif, écrire les solutions $(- b \pm \sqrt{\Delta})/2a$.
Au prix d'un travail mathématique profond, Galois montre que les seules équations ainsi résolubles sont celles dont les racines satisfont certaines propriétés de structure, généralement absentes à

partir du degré 5. Au-delà, pour résoudre une équation, inutile de chercher une recette consistant à combiner ses coefficients : il est démontré qu'il n'y en a pas.

Pour ce qui est de la question générale de savoir si un problème donné possède une solution algorithmique, l'asymétrie entre les réponses OUI et les réponses NON prend la tournure suivante. Si vous avez trouvé un algorithme pour résoudre votre problème, c'est qu'il en existe un. En revanche, pour établir qu'il n'y a *pas* d'algorithme pour exécuter une tâche donnée, l'intuition ne suffit pas. Il faut formaliser les choses. Ceci est assez normal, si l'on songe qu'un résultat d'indécidabilité ou d'incalculabilité algorithmique a la portée universelle d'un principe de thermodynamique : aucune évolution technologique ne donnera ni le mouvement perpétuel ni la résolution par algorithme d'un problème démontré insoluble par ce biais.

Le travail conséquent qui a permis d'établir que certains problèmes n'ont pas de solution algorithmique a été effectué au milieu des années 1930. Au terme de l'année 1936, *annus mirabilis* pour la théorie des algorithmes, nous sommes dans la situation suivante. D'une part, on a trouvé une caractérisation « absolue » de la notion de calculabilité par algorithme. Un grand nombre de logiciens éminents essaient, à peu près au même moment, de caractériser la notion d'algorithme, et ils s'y prennent tous de manières très différentes (la manière aujourd'hui la plus connue, celle du mathématicien britannique Alan Turing, n'en est qu'une parmi d'autres). Or, les fonctions calculables par les algorithmes ainsi diversement caractérisés s'avèrent les mêmes, un peu comme des billes qui, lâchées de divers endroits du bord d'un bol, s'y retrouvent groupées au fond. Il y a là une sorte d' « attracteur » : les logiciens de l'époque ont bel et bien vu le même objet, mais sous des perspectives différentes. Cette convergence, jointe à d'autres arguments techniques, montre qu'on a enfin mis la main sur une notion robuste et stable d'algorithme (une situation que Gödel devait qualifier de « miraculeuse »).

D'autre part, en 1936, on a démontré l'existence de fonctions non calculables par algorithmes (ou, de façon équivalente, de problèmes insolubles par algorithmes). S'agissant des fonctions de **N** dans **N** - car c'est toujours d'elles dont il s'agit lorsqu'on parle de calculabilité, du moins à cette époque -, l'argument qui établit l'existence de ces fonctions peut résulter d'un simple argument de cardinalité : les algorithmes (ce que nous appellerions aujourd'hui les « programmes ») sont des textes finis, dont on peut écrire la liste. Alors que les fonctions de **N** dans **N** forment un ensemble infini de cardinalité plus haute (il y en a autant que de nombres réels, or les nombres réels ne peuvent être énumérés comme le sont les entiers).

Mais un argument donné par le mathématicien britannique Alan Turing lui-même prend un tour plus précis et « constructif ». Selon lui, il y a une question très particulière qui ne peut être résolue à

l'aide d'un algorithme : celle de l' « arrêt des machines de Turing » (lire encadré 1), qui consiste justement à se demander si un algorithme donné va fournir ou non un résultat pour un argument donné. C'est en démontrant cette impossibilité particulière que Turing établira que certains problèmes, parfaitement définis, n'admettent aucune solution algorithmique (lire encadré 2).

On pourrait à l'infini discuter la signification « philosophique » de ce résultat, qui montre les limites des algorithmes mais, aussi bien, l'étendue de leur empire : des machines qui les incarnent, naissent des problèmes qui en excèdent les capacités de décision. Mieux vaut, pour conclure, signaler que la période « classique » de la théorie des algorithmes, ouverte par Turing, n'est plus vraiment la nôtre, et ce pour une raison essentielle : la frontière de Turing, celle qui sépare le calculable de l'incalculable, n'est plus vraiment la nôtre non plus.
Compte tenu de l' « incarnation » des algorithmes dans des machines physiques et de la place désormais tenue par ces machines dans nos vies, ce qui nous intéresse n'est plus de savoir si un problème a une solution algorithmique, mais quelle est la complexité de l'algorithme en question ou sa rapidité d'exécution. Il existe en principe un algorithme permettant de gagner à coup sûr aux échecs si l'on joue avec les Blancs, et il en existe un autre qui permet à un voyageur de commerce d'optimiser la longueur de ses tournées. Mais ces algorithmes, dont le temps d'exécution est une fonction exponentielle de la taille des données auxquelles on les applique, ne font pas l'affaire pour les créatures pressées que nous sommes : de l'algorithme qui établit la position souhaitable des parties mobiles des ailes d'avion en fonction de sa vitesse, de sa position relative au sol, etc, nous voulons, non seulement qu'il s'arrête avec un résultat pour les valeurs données, mais qu'il délivre son verdict *avant* que les roues ne touchent le sol. En outre, loin des machines abstraites imaginées par Turing, le calcul a un coût : à quoi nous servirait, pour économiser l'argent que nous gaspillerions en choisissant un trajet routier sub-optimal, de nous engager dans un calcul d'optimalité qui nous coûterait plus que l'économie réalisée ?

ENCADRE.

Les machines de Turing.

Pour analyser ce que « calculer de façon algorithmique » veut dire, Turing a introduit en 1936 la notion de « machine » qui porte désormais son nom, et qui est une représentation idéalisée d'un homme qui calcule. Un tel automate abstrait comporte, d'une part un ruban infini divisé en cellules pouvant contenir des 0 ou des 1, d'autre part une tête positionnée face au ruban et capable de se déplacer d'une cellule à l'autre dans les deux sens (cette tête est en outre capable d'être dans divers états internes, dont un état initial et un état final).
Un algorithme est, dans ce format, représenté par la suite finie des « instructions » de la machine, une instruction <q,S,X,q',S'> étant une consigne du type « si, dans l'état q, vous êtes en face

d'une cellule contenant le symbole S (donc un 0 ou un 1), alors faites le mouvement X (déplacez vous d'un pas vers la gauche, ou vers la droite, ou ne bougez pas), passez dans l'état q' et écrivez le symbole S' dans la cellule qui est en face de vous ». On dit que la machine M calcule la fonction *f* si, chaque fois que, dans son état initial, elle contemple le premier élément d'une suite de *n* 1's, elle achève son travail en étant dans son état final devant le dernier élément d'une suite de *f(n)* 1's.

Noter que la notion d'état interne d'une machine de Turing a été souvent tenue pour une explication idéalisée plausible de ce que pourrait être un « état mental » d'un agent humain : la réaction que nous avons en face d'un stimulus (ici, le symbole que nous avons en face de nous) dépend, non seulement de la nature de ce stimulus, mais de notre état interne au moment où nous le contemplons.

En encadré : la preuve de Turing

Selon Turing la question de l'arrêt des machines de Turing, qui consiste à se demander si un algorithme donné va fournir ou non un résultat pour un argument donné, ne peut être résolue à l'aide d'un algorithme. En voici la preuve, inspirée de la méthode de « diagonalisation » utilisée par Cantor pour démontrer l'impossibilité d'énumérer l'ensemble des nombres réels. Supposons une liste $f_0, f_1, \ldots$ de toutes les fonctions calculables, et définissons la fonction *g* dont la valeur pour *n* est égale à $f_n(n)+1$ si f_n est définie pour *n*, et à 0 sinon. Cette fonction ne figure certainement pas dans la liste, puisqu'elle diffère de la *n*-ième fonction de cette liste par sa valeur pour *n*, et n'est donc pas une fonction calculable. Or cette incalculabilité ne peut provenir que de la clause « *g(n)*=0 si f_n n'est pas définie pour *n* ». Il ne saurait donc exister d'algorithme capable de décider si un algorithme donné va s'arrêter avec un résultat pour un opérande donné.

Pour en savoir plus

Histoire de l'Académie Royale des Sciences (Année 1775), Paris, Imprimerie Royale, 1778 p 61 sq.

http://bit.ly/1ToliEg