informatique

la démarche algorithmique dans l'enseignement (*)

par G. Rauzy,
IREM de Marseille

Dans ce bulletin, sont déjà parus d'autres articles sur l'apport éventuel de l'informatique à l'enseignement des mathématiques : utilisation en tant qu'outil, au même titre que l'audiovisuel par exemple, idées d'algorithmes, numériques notamment, situations où l'élève expérimente ou programme lui-même, etc.

Je voudrais insister, ici, sur l'intérêt de l'approche algorithmique en mathématiques. Ce dont personne évidemment ne doute, mais que les programmes actuels ou récents ont tendance à ignorer, au profit d'autres approches, souvent plus élégantes et non moins fructueuses, certes, mais qui ne sont cependant ni meilleures, ni moins bonnes, même si à certains stades du développement des mathématiques, elles ont pu être privilégiées.

Cette disparition progressive de la démarche algorithmique dans l'enseignement a certainement de multiples raisons, et notamment, la difficulté d'en abstraire la généralité, sans tomber dans un formalisme préalable excessif.

(*) Conférence du 25.10.85 aux Journées Nationales de Port-Barcarès.

Entendons nous bien ! Il ne paraît pas souhaitable de faire en tant que tel, un enseignement d'algorithmique au Collège ou au Lycée. Simplement, il s'agit d'écrire et de faire écrire quelques algorithmes, en tirant profit, sur ce plan, de l'apport conceptuel de l'informatique qui, avec un minimum d'arsenal technique, permet de les décrire, d'une manière plus "naturelle" en un sens, et en tout cas, plus abordable que nos antiques "et ainsi de suite...", "etc..." ou tout autre variante plus ou moins heureuse.

Je me propose donc, de préciser cette description sur quelques exemples, et de montrer comment, un algorithme étant conçu comme **une règle de succession de "situations" qui fait passer d'une situation initiale donnée à une situation finale recherchée**, il est possible de "faire la preuve" que le résultat obtenu est bien celui qu'on souhaitait.

Ce qui semble la moindre des choses ! Mais, n'est pas évident : qui est vraiment convaincu, sauf à refaire l'élévation au carré dans chaque cas, de la justesse de l'algorithme que l'on pratiquait jadis "à la main" pour l'extraction d'une racine carrée ?

Premier exemple

Nous partons d'une transformation T définie sur des triplets (a,b,p) d'entiers positifs ou nuls, définie comme suit :

1) Si b est impair, p est transformé en $p+a$. Sinon, p reste inchangé.

2) a et b sont respectivement transformés en $2a$ et en q quotient de la division de b par 2.

Observons que l'ordre dans lequel sont effectuées ces opérations est important : partant, par exemple, du triplet (3, 7, 1), T le transforme en (6, 3, 4) alors qu'en appliquant 2 puis 1, on obtiendrait (6, 3, 7).

Remarquons également que si b est au départ strictement positif, sa nouvelle valeur après transformation sera strictement inférieure à l'ancienne.

Nous pouvons, maintenant, décrire notre algorithme. Partons d'une situation initiale où $a = a_0$, $b = b_0$ et $p = 0$.

Si b est différent de 0, appliquons à ce triplet la transformation T, et recommençons de même sur le nouveau triplet, et,... ainsi de suite... tant que b reste différent de zéro.

En vertu de la remarque faite précédemment, nous obtiendrons au bout d'un certain temps, un triplet (a_1, b_1, p_1) où b_1 sera nul. Nous arrêtons alors le calcul, et convenons de dire que son résultat est p_1.

Cette description forcément un peu longue en langage courant se traduit aisément en principe, dans n'importe quel langage de programmation, et avec un minimum d'habitude de façon très claire.

Dans un semi-langage, nous dirons par exemple :

"**Tant que** $b \neq 0$ **faire,**
si b **impair, alors,** $p \longleftarrow p + a$**. fin.**
$a \longleftarrow 2a$, $b \longleftarrow$ **quotient de** b **par 2.**
fin"

(La flèche $x \longleftarrow y$ signifiant que le nombre désigné par la lettre x, la "variable" dira-t-on, "reçoit" comme nouvelle valeur celle attribuée à y. On écrit aussi $x := y$ ou, hélas !, $x = y$).

Demandons nous maintenant, ce que calcule cet algorithme, c'est-à-dire, que vaut p_1 en fonction de a_0 et b_0.

L'observation fondamentale est ici, que **lorsqu'on effectue la transformation T, la quantité** $ab+p$ **reste inchangée**, comme on le vérifiera aisément.

Elle aura donc la même valeur en fin de calcul qu'au début de celui-ci. Valant a_0b_0 initialement puisque, p_0 est nul, et p_1 à la fin puisque, cette fois b_1 est nul, il en résulte que :

$$p_1 = a_0b_0$$

Nous avons ainsi calculé le produit des nombres a_0 et b_0, par une méthode qui remonte d'ailleurs, à la plus haute antiquité !

Premier commentaire

Nous avons donc là, une manière de nous assurer qu'un algorithme effectue bien un calcul donné : **chercher une quantité qui reste inchangée tout au long des étapes successives du calcul, et comparer alors les formes qu'elle prend au début du calcul et à la fin dudit.**

Cette notion n'est pas nouvelle, certes, elle est même absolument fondamentale dans la démarche mathématique : cela s'appelle "chercher des invariants".

Malgré son importance, cependant, elle n'a fait que de timides apparitions, souvent fugitives, dans notre enseignement : il serait trop long ici de tenter d'en cerner les raisons, comme de faire une véritable analyse didactique de son obsolescence, en géométrie notamment (ou bien, est-ce la géométrie, elle-même, qui...?).

Excellente occasion donc, que de la mettre en évidence par le biais d'algorithmes !

Commentaire sur le précédent

Mais, dira-t-on, l'invariant mis en évidence sur l'exemple précédent, explique peut-être, pourquoi l'algorithme calcule bien ce que l'on veut, mais semble parfaitement "parachuté" : s'il est vrai que c'est un "truc" commode, comment faire en sorte que ce ne soit pas seulement un truc, à réinventer précisément dans chaque cas particulier ?

Remarquons en passant, qu'il en est souvent ainsi au niveau de la recherche mathématique : on trouve sans cesse de nouveaux invariants plus fins ou mieux adaptés à ce que l'on veut identifier ou différencier.

Par contre, ici, l'approche est différente et en quelque sorte inversée, bien que cela demande à être nuancé : nous ne cherchons pas à distinguer ou au contraire rassembler des objets existant a priori. Nous cherchons à construire ces objets, situations, algorithmes, programmes...

Et, nous allons **les bâtir autour de la notion d'invariant**, comme une trame sur laquelle nous tisserons notre canevas.

Exemple de tissage

Je veux, étant donnés deux entiers positifs x et n (x non nul), calculer x^n, en effectuant le moins possible de multiplications.

Quelques exemples du type qui précède m'ont permis de remarquer l'intérêt de la division par 2 dans certaines circonstances.

Je tente donc cette division sur x ou sur n, ou sur les deux à la fois : je saisis plus ou moins vite, qu'il vaut mieux essayer sur n...

J'écris alors : $n = 2q + r$ r valant 0 ou 1
et obtiens : $x^n = (x^2)^q \times x^r$

Je suis presque ramené à une situation plus "simple" : calculer une puissance d'un nombre plus grand, sans doute (c'est pour cela que je l'ai appelé x, avec son pouvoir évocateur d'inconnu), mais avec un exposant plus petit q au lieu de n. Ne nous attardons pas sur le plus "simple" : ce n'est pas encore de la récurrence, c'est déjà de la récursivité...

A ceci près cependant, que le second membre de mon égalité n'est plus tout à fait de la même forme que le premier : j'avais une puissance, j'ai un monôme.

C'est là que se situe un pas délicat, où seule va me guider l'idée de "situation générale" : plutôt que de calculer seulement une puissance, je vais calculer un monôme. La difficulté est du même ordre en géométrie; quand on dit : "menons la perpendiculaire issue de A sur BC et appelons H le point d'intersection...". Je n'en suis pas sûr : le concept de situation générale, une fois acquis, pousse à généraliser, précisément. Il reste encore un éventail de possibles : du moins, la problématique a-t-elle gagné en clarté !

Je généralise donc, bravement, en introduisant une nouvelle variable z et cherche à calculer le produit $z \times x^n$

Toujours avec : $n = 2q + r$
j'obtiens : $z \times x^n = (z \times x^r) \times (x^2)^q$

Ce qui me "prouve" que la quantité $z \times x^n$ reste invariante par la transformation suivante :

1) Si n impair, alors, $z \longleftarrow zx$. fin.
2) $x \longleftarrow x^2$, $n \longleftarrow q$

(NOTA : quelques expériences malheureuses, mais hors sujet, prouvent si l'on ne veut pas mélanger les genres, qu'il vaut mieux écrire *"$x \longleftarrow x \times x$" que "$x \longleftarrow x^2$"*. Cela sera sous-entendu, par la suite).

Je peux maintenant écrire l'algorithme de calcul de x^n, étant entendu que je dois donner à z la valeur 1 dans la situation initiale, sa production finale étant obtenue lorsque l'invariant est égal à la dernière valeur de z calculée, c'est-à-dire quand cette fois, x^n est égal à 1, soit encore, x étant strictement positif, si n est égal à 0.

On écrira donc :

"$z \longleftarrow 1$
Tant que $n \neq 0$ faire
si, n impair, alors, $z \longleftarrow zx$. fin.
$x \longleftarrow x^2$, $n \longleftarrow$ quotient de n par 2.
fin"

La similitude de cet algorithme avec le précédent est frappante : je laisse au lecteur le soin d'imaginer une situation plus générale...

Hors propos

En écrivant l'entier n en base 2, et remarquant que, à chaque étape, la longueur de l'écriture correspondante diminue d'une unité, on voit que l'algorithme précédent exige au plus $2 \times \ell(n)$ multiplications, $\ell(n)$ désignant le nombre de chiffre en base 2 de n (on pourrait être plus précis puisque l'une des multiplications est "commandée" par la condition d'imparité de n). En fait, quitte à l'améliorer en l'écrivant autrement (la première multiplication de z par x est triviale, la dernière élévation au carré de x ne sert à rien), on arrive à $2\,\ell(n) - 2$ multiplications : ainsi x^{1023} se calculera avec 18 multiplications seulement, ce qui est donc, une réponse partielle à notre exigence initiale de minimiser le nombre de multiplications.

Observons que ce n'est pas la meilleure, même si l'on a cherché à la préciser. Avec $n = 15$ par exemple, cela exige 6 multiplications, alors que l'algorithme :

"$y \longleftarrow x \times x$, $y \longleftarrow x \times y$, $z \longleftarrow y \times y$, $y \longleftarrow y \times z$, $y \longleftarrow y \times z$"

aboutit à la valeur $y = x^{15}$ en 5 multiplications seulement.

Si un dispositif analogique quelconque devait répéter un grand nombre de fois cette élévation à la puissance 15, le gain de temps serait dans le rapport de 5 à 6, ce qui est loin d'être négligeable...

Mais nous avons quand même trouvé une "bonne" solution dans le cas général, où n est quelconque, sans qu'elle soit nécessairement la "meilleure" : on s'en contentera dans toute situation réelle où d'autres impératifs de temps d'étude et de rentabilité vont intervenir, du moins tant qu'une nouvelle amélioration théorique ne devient pas "indispensable"...

Avec d'autres critères de valeur, nous nous retrouvons dans la situation du chercheur en mathématiques face à la découverte... de nouveaux invariants, par exemple. Mais, notre ambition n'étant pas de transformer tous nos élèves en futurs mathématiciens professionnels, il ne me semble pas inutile de souligner quand on en a l'occasion (et, au besoin la créer...) qu'un problème, ou plutôt une problématique n'a pas généralement, une seule "bonne" solution, mais une approche plus ou moins heureuse, où les connaissances que nous visons à enseigner peuvent jouer un rôle déterminant, y compris dans les questions qu'elles soulèvent.

Dernier exemple

J'ai passé sous silence dans l'algorithme précédent, le fait que celui-ci, aboutissant lorsque n vaut 0, il faut qu'à ce moment là, la valeur de x soit non nulle (pour éviter la forme pathologique 0^0), et qu'intermédiairement d'ailleurs, toute étape du calcul préparait à une autre où les opérations sont "exécutables", c'est-à-dire où le nouveau triplet (z, x, n) reste dans le domaine de définition de la transformation envisagée.

C'était évident, mais cela ne l'est pas nécessairement dans un cadre plus général, et, cette évidence cachait un fait essentiel : l'invariant "zx^n" qui a permis de conclure, s'alliait sous forme implicite à la constatation que, au cours du calcul, il restait constamment défini. Bref, c'était plutôt une assertion logique énonçant, par exemple : **"z, x et n sont des entiers positifs, x est strictement positif et zx^n reste inchangé".**

Ce n'est pas couper les cheveux en quatre, car cela va être presque toujours le cas. Mais, il n'y a pas lieu de s'effrayer : **l'invariant sera, simplement, en général, une phrase du langage courant qu'il est inutile, à ce stade, de formaliser plus avant**, et qui y gagnera même en spontanéité !

Donnons en, un exemple typique qui, bien qu'écrit avec des nombres, se rattache déjà à des algorithmes "non numériques". Il s'agit de la recherche d'un mot dans un dictionnaire, à caractère dichotomique pour faire le lien avec ce qui précède, et, évoquant de manière discrète (en tant qu'opposé à continu), le bien connu "théorème des valeurs intermédiaires".

Le problème est le suivant :

"Soit $a(1)<a(2)<\ldots<a(n)$ une suite strictement croissante de nombres (entiers, si l'on veut) et x un nombre vérifiant :

$$a(1)\leqslant x<a(n)$$

Le but du calcul est de déterminer l'entier j tel que :

$$a(j) \leqslant x < a(j+1)$$ "

Passons sur les essais. La conscience d'une recherche de "situation générale" amène à définir celle-ci, comme la situation suivante :

"u et v sont deux entiers tels que $1 \leqslant u < v \leqslant n$ et $a(u) \leqslant x < a(v)$"

Au départ, la situation est $u = 1$ et $v = n$
A la fin, elle sera $u = j$ et $v = j+1$

Il faut donc chercher à réduire au cours des étapes successives l'intervalle $[u, v]$.

L'invariant est tout trouvé : c'est précisément la proposition définissant la situation générale, affirmée comme restant vraie à chaque étape.

La méthode de passage d'une situation à la suivante dépend de notre degré de connaissance ou d'invention. Voulant illustrer la méthode dichotomique nous écrirons donc, l'algorithme suivant dont le résultat est u :

" $u \leftarrow 1$, $v \leftarrow n$
Tant que $u+1 < y$ **faire**
$q \leftarrow$ **partie entière de** $\frac{u+v}{2}$
si, $a(q) \leqslant x$, **alors,** $u \leftarrow q$
sinon, $v \leftarrow q$. **fin.**
fin"

Pour conclure

Nous nous sommes volontairement limités, ici, à un champ bien précis d'exemples, illustrant une méthode classique "d'accélération" d'algorithmes. D'autres choix sont évidemment possibles et souhaitables avec d'autres objectifs sous-jacents qui, à tel ou tel niveau, ne demandent pas à être théorisés : l'itération comme recherche éventuelle de points fixes ou périodiques, la réversibilité de certains algorithmes, l'utilisation d'une énumération pour établir certaines formules de récurrence, la traduction automatique d'un algorithme non-déterministe en algorithme déterministe par "effet miroir" etc.

Il était évidemment impossible de tracer toutes les directions possibles et a fortiori de les concrétiser en exemples suggestifs : c'est un travail à faire, en grande partie, et cela pourrait l'être notamment dans les IREM.

Enfin, je n'ai pas envisagé la question de "sortie de la boucle", c'est-à-dire de savoir si un algorithme s'arrête à un moment donné ou continue indéfiniment, les exemples pris aboutissant presque évidemment. C'est pourtant là une question fort délicate. Qu'on en juge par l'algorithme sui-

vant, où l'on ne sait toujours pas si pour toute valeur de l'entier n il s'arrête ou continue indéfiniment :

" Tant que $n > 1$ faire
si, n pair, alors, $n \leftarrow n/2$
sinon, $n \leftarrow 3n + 1$. fin.
fin ".

NOTA : On consultera avec profit le livre de J. ARSAC "Les bases de la programmation" (Dunod Informatique - 1983), sans compter les diverses éditions de D. KNUTH "The Art of Computer Programming" (Addison Wesley) et, puisqu'il ne s'agit pas d'une bibliographie exhaustive,... la prochaine brochure à paraître de l'IREM d'Aix-Marseille.