Université d'Aix-Marseille

2013-2014

Master « Mathématiques et applications »

Spécialité « Enseignement et formation en mathématiques »

Parcours « Didactique » (2e année)

UE 1

Fondements et méthodes de la recherche en didactique

Yves Chevallard & Michèle Artaud

TD 2. Analyse critique d'un article

Une équipe de recherche Ξ_2 envisage d'élaborer un programme de recherche en didactique de l'algorithmique au secondaire. On suppose que, au départ, Ξ_2 n'est en rien familière avec l'algorithmique et avec son enseignement à quelque niveau que ce soit. Pour amorcer le travail projeté, Ξ_2 consacre une partie de son séminaire à l'étude de travaux existants qui pourraient relever de ce secteur de la didactique. L'équipe Ξ_2 s'arrête d'abord sur un article appelé ci-après « document *E* » (dont on trouvera les références un peu plus loin), à propos duquel elle se pose la question suivante :

> Q^2_∂. En quoi et dans quelle mesure le document *E* expose-t-il une recherche en didactique des mathématiques ?

☞ *Dans tout ce qui suit dans ce TD 2, on se placera dans la perspective de l'étude de la question Q^2_∂ ci-dessus.*

Question 1. L'équipe Ξ_2 décide d'enquêter sur les auteurs du document *E* et d'analyser en un tout premier temps le résumé et la conclusion de l'article. Que peut-elle ainsi mettre au jour ? Quelles interrogations cela soulève-t-il ?

Données 1.1

☛ Fiche signalétique du document *E*

- Auteur : Simon Modeste, Sylvain Gravier et Cécile Ouvrier-Buffet.
- Titre : Algorithmique et apprentissage de la preuve.
- Revue : *Repères - IREM*
- Volume : 79 (avril 2010)
- Pages : 51-72
- Référence APA adapté au français : Modeste, S., Gravier, S. & Ouvrier-Buffet, C. (2010). Algorithmique et apprentissage de la preuve. *Repères - IREM*, *79*, 51-72.

Éléments d'analyse 1.1

1. Le premier auteur, *Simon Modeste*, est docteur en didactique des mathématiques (voir http://www-fourier.ujf-grenoble.fr/~modeste/). Sa thèse, dirigée par Sylvain Gravier et Cécile Ouvrier-Buffet, est intitulée *Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ?* Elle a été soutenue le 5 décembre 2012 et est disponible en ligne à l'adresse http://tel.archives-ouvertes.fr/tel-00783294. Le deuxième auteur, Sylvain Gravier (http://mathsamodeler.ujf-grenoble.fr/~sgravier/SG.htm), est mathématicien, directeur de recherche au CNRS, spécialiste de théorie des graphes et de géométrie discrète et est responsable de l'équipe « Maths à modeler » à l'Université Joseph-Fourier (Grenoble). Cécile Ouvrier-Buffet (http://www.lar.univ-paris-diderot.fr/user/91) est maître de conférences de mathématiques à l'ESPE de l'Université Paris-est Créteil (ex-Université Paris 12), didacticienne des mathématiques, responsable du groupe « Démarches de recherche en mathématiques » à l'IREM de l'Université Paris 7.

2. On a affaire à l'origine à une configuration de recherche tout à fait classique, qui se concrétise par la formation d'un *système d'étude et de recherche* qu'on peut écrire ainsi : $S(\xi_0 ; \Xi ; \mathcal{Q}_0)$, où ξ_0 est le doctorant Simon Modeste et $\Xi = \{\xi_1, \xi_2\}$, ξ_1 étant le directeur de thèse Sylvain Gravier et ξ_2 la co-directrice de thèse Cécile Ouvrier-Buffet, le symbole $\mathcal{Q}$ désignant l'ensemble de questions – éventuellement réduit à une question – sur lequel a porté le travail de thèse de ξ_0. Au cours du fonctionnement de $S(\xi_0 ; \Xi ; \mathcal{Q})$ se forme un système d'étude et de recherche qu'on doit écrire formellement $S(\{\xi_0, \xi_1, \xi_2\} ; ? ; \mathcal{Q}_1\}$, où $\mathcal{Q}_1$ est un certain complexe de questions qui, à ce stade de l'analyse, *reste à préciser*. Ce système produit alors une proposition d'article porteuse d'une réponse $\mathcal{R}_1^{\heartsuit}$ à $\mathcal{Q}_1$.

3. Le titre de l'article – « Algorithmique et apprentissage de la preuve » – associe deux entités : d'une part, le domaine de l'algorithmique ; d'autre part, celui de ce qu'on appelle habituellement, en français, « l'apprentissage de la démonstration ». Ni la nature de cette association, ni ce que l'article établirait à son propos n'est explicite dans le titre choisi, ce qu'on pourrait éventuellement attendre s'agissant du titre d'un article de recherche. On notera l'usage du mot *preuve* en lieu et place du mot *démonstration*. Cela est-il l'indice d'un léger « déplacement » de la notion de démonstration mise en jeu dans cet article ? Ou bien cela témoigne-t-il du fait que, d'une manière ou d'une autre, les auteurs évoluent dans un monde où « preuve » tend à se substituer à « démonstration », par exemple sous l'influence de l'anglais *proof* ? Notons que le mot anglais *demonstration* s'emploie également, en anglais, pour désigner *a proof*, même si ce mot à d'autres emplois plus habituels (à savoir « manifestation [de rue] » et « démonstration [d'appareils ménagers, etc.] », emplois que recense la notice suivante, extraite du dictionnaire en ligne *Wordreference* (voir à l'adresse http://www.wordreference.com/enfr/demonstration) :

demonstration /ˌdemənˈstreɪʃən/

English definition | English synonyms | in Spanish | conjugator | in context | images

WordReference | Collins

WordReference English-French Dictionary © 2014:

Principal Translations/Principales traductions

demonstration *n*	(showing sthg)	démonstration *nf*
demonstration *n*	(public)	manifestation *nf*

Is something important missing? Report an error or suggest an improvement.

WordReference English-French Dictionary © 2014:

Principal Translations/Principales traductions

demo, demonstration *n*	*informal, abbr* (protest march) *familier, abréviation de : manifestation*	manif *nf*
demo, demonstration *n*	*informal, abbr* (demonstration of product or technique) *familier, abréviation de : démonstration*	démo *nf*

Is something important missing? Report an error or suggest an improvement.

Notons, par exemple, le titre que James Joseph Sylvester (1814-1897) a donné à l'article où il démontrait sa « loi d'inertie » (1852) : « A demonstration of the theorem that every homogeneous quadratic polynomial is reducible by real orthogonal substitutions to the form of a sum of positive and negative squares ». (Cet article a été publié dans le *Philosophical Magazine*, Ser. 4, vol. 4, n° 23, pp. 138-142 ; on le retrouvera en ligne à l'adresse suivante : http://www.maths.ed.ac.uk/~aar/sylv/inertia.pdf.) Ce titre, qui serait regardé sans aucun doute

comme *trop long* aujourd'hui, illustre en revanche clairement l'usage de préciser le ou les résultats qu'on tente d'établir dans l'article que titre annonce.

3. La revue *Repères IREM* où est publié l'article est « la revue des IREM ». C'est une revue trimestrielle qui paraît depuis 1990 et s'adresse préférentiellement aux professeurs, formateurs et autres personnels impliqués dans l'enseignement des mathématiques sans être pour autant des chercheurs en didactique des mathématiques. Un article publié dans cette revue n'est donc pas censé être à tout coup un article de recherche en didactique des mathématiques. Il peut fort bien ne pas être un article *de recherche*. S'il l'est, il peut ne pas être un article de recherche *en didactique*. Et, s'il s'agissait bien d'un article de recherche en didactique, il se pourrait encore que ce ne soit pas de la didactique *des mathématiques*… La question Q_{∂}^{2} posée plus haut à propos de l'article est donc judicieuse.

4. L'article occupe 22 pages de la revue qui le publie. Il a sans doute un lien organique avec la thèse de Simon Modeste, laquelle n'a cependant été soutenue qu'*après* la publication de l'article.

Données 1.2

☛ Résumé de l'article (p. 51)

RÉSUMÉ

Dans cet article, nous présentons tout d'abord une étude épistémologique sur la place et le rôle de l'algorithme dans la science mathématique. Nous étudions les différents aspects de l'algorithme suivant une dichotomie outil-objet, puis nous développons le lien privilégié qu'il entretient avec la preuve. En s'appuyant sur cette étude, nous proposons une analyse des programmes du lycée ainsi que des manuels. Nous proposons, dans un troisième temps, une situation de recherche en classe mettant en jeu l'algorithme. Les résultats d'expérimentations de cette situation montrent comment la construction d'algorithmes, leur preuve et l'analyse de leur complexité peuvent être questionnées en classe.

Éléments d'analyse 1.2

1. Un premier point à noter concerne le vocabulaire employé. Les auteurs parlent en effet de *l'*algorithme pour signifier, apparemment, « *les* algorithmes », « *l'algorithmique* » ou « *la notion* (ou : *le concept*) *d'algorithme* ». Imaginons, en lieu et place de la déclaration initiale (« Dans cet article, nous présentons tout d'abord une étude épistémologique sur la place et le rôle *de l'algorithme* dans la science mathématique »), la formulation « Dans cet article, nous présentons tout d'abord une étude épistémologique sur la place et le rôle *du nombre* dans la science mathématique » ou encore « Dans cet article, nous présentons tout d'abord une étude épistémologique sur la place et le rôle *du graphe* dans la science mathématique », évoquant ensuite « les différents aspects *du nombre* suivant une dichotomie outil-objet » ou « les différents aspects *du graphe* suivant une dichotomie outil-objet » avant de mentionner « une situation de recherche en classe mettant en jeu « *le nombre* » ou « *le graphe* ». On aura noté que le titre – « Enseigner l'algorithme pour quoi ? » – de la thèse de Simon Modeste, codirigée par Sylvain Gravier et Cécile Ouvrier-Buffet – les deux autres signataires de l'article examiné –, utilise semblablement le singulier générique (qu'on emploi lorsqu'on évoque « l'homme », « l'animal », etc.). Doit-on voir dans cet usage, ainsi qu'on l'a suggéré plus haut, une ellipse de l'expression « la notion de » (ou « le concept de »), « l'algorithme » signifiant alors « la notion d'algorithme », « le nombre » se substituant à « la notion de nombre », « le graphe » renvoyant à « la notion de graphe » ? Notons que si, en anglais, on parle bien de *graph theory*, de *group theory*, de *number theory*, etc., on parle usuellement, en français, de « théorie *des* graphes », et non de « théorie *du* graphe », de « théorie *des* groupes », et non de « théorie *du* groupe », de « théorie *des* nombres », et non de « théorie *du* nombre », etc. Quel est alors le sens de l'emploi du singulier par les auteurs ? Cet emploi a-t-il une incidence sur l'étude présentée ou sur sa présentation ? Autant de questions qui, à ce stade de notre étude de l'article, restent ouvertes.

2. À condition de laisser de côté l'*introduction* (pp. 51-52), la *conclusion* (p. 71) et les *références bibliographiques* (p. 72), le résumé découpe l'article en trois parties. Tout d'abord vient une partie 1 intitulée *L'algorithme, un objet qui questionne l'activité mathématique et l'enseignement* (pp. 53-57). Le résumé la présente comme « une étude épistémologique sur la place et le rôle de l'algorithme dans la science mathématique », où les auteurs examinent « les différents aspects de l'algorithme suivant une dichotomie outil-objet » avant de développer « le lien privilégié qu'il entretient avec la preuve ». (Notons, là encore, l'usage du singulier générique : « *la preuve* »). Vient ensuite la partie 2, intitulée *L'algorithme dans*

l'enseignement : outil ou objet ? (pp. 57-60), où les auteurs, prenant appui sur la partie 1, proposent « une analyse des programmes du lycée ainsi que des manuels ». Enfin l'article comporte une partie 3 intitulée *Une situation pour l'algorithme* (pp. 60-71), de loin la plus longue, où les auteurs explicitent « une situation de recherche en classe mettant en jeu l'algorithme ».

3. Les expérimentations réalisées, soulignent les auteurs, « montrent comment la construction d'algorithmes, leur preuve et l'analyse de leur complexité peuvent être questionnées en classe ». Cela promet au lecteur que la situation étudiée met les élèves aux prises avec des aspects problématiques qui renvoient, respectivement, à la notion de *construction d'un algorithme*, de *preuve d'un algorithme* et de *complexité d'un algorithme*. (Notons que, de ces trois notions, le titre de l'article – *Algorithmique et apprentissage de la preuve* – ne retient explicitement que la notion de *preuve*.) Si cette conclusion est bien vérifiée, elle signifie que la situation proposée et expérimentée est *très riche*. Corrélativement, on peut imaginer que sa gestion didactique par les élèves et le professeur est sans doute *délicate* (ce qui, à ce stade, n'est, bien entendu, qu'une conjecture).

Données 1.3

☛ Conclusion de l'article (p. 71)

Conclusion

L'algorithme peut être vu sous différents aspects, dont certains ont un potentiel réel pour l'enseignement des mathématiques et en particulier pour l'apprentissage de la preuve. Malheureusement, ces aspects semblent absents des programmes et manuels de l'enseignement secondaire en France et sont donc difficiles à exploiter par les enseignants. En effet, l'algorithme ne joue dans les programmes et les manuels que le rôle d'un outil. Pourtant, il semble que l'on peut questionner les différents aspects de ce concept dans certaines situations, et tenter de faire apparaître l'algorithme comme un véritable objet mathématique.

En proposant le « problème des pesées » à des étudiants, sous la forme d'une situation de recherche, nous avons pu constater que la construction d'algorithmes, leur preuve et leur étude pouvaient vivre en classe. Cependant, l'étude de leur complexité reste une question difficile. La complexité est une notion spécifique à l'algorithmique (Knuth, 1985) et son étude passe nécessairement par l'objet algorithme. Cette notion pourrait trouver sa place au sein d'un enseignement de mathématiques et même redonner du sens à certains des objets enseignés.

L'algorithme est sur le point de prendre une place plus grande dans l'enseignement français et il est nécessaire de suivre cette évolution et de l'accompagner.

Éléments d'analyse 1.3

1. Le point le plus frappant sans doute est le fait que l'algorithmique – « l'algorithme », comme l'écrivent les auteurs – serait intéressante, *non pas « en soi et pour soi »*, mais pour ce qu'elle *pourrait* apporter à *l'enseignement des mathématiques*. Certains « aspects » de l'algorithmique, concluent les auteurs, auraient « un potentiel réel pour l'enseignement des mathématiques ». C'est assigner à l'algorithmique un rôle de « faire-valoir » et de « remontant » des mathématiques, à cause de certaines de ses « vertus » supposées, et notamment à cause du rapport privilégié qu'elle entretiendrait avec une question censée traditionnellement occuper le cœur du mathématique : la question de « la preuve » – de la démonstration. La fonction ancillaire ainsi dévolue à « l'algorithme » pèse-t-elle sur l'inscription du contenu du document *E* dans le champ de la recherche en didactique des mathématiques ?

2. L'apport potentiel de l'algorithmique à l'enseignement des mathématiques reste cependant peu reconnu et peu intégré, ajoutent les auteurs, parce que ses aspects pertinents « semblent absents des programmes et manuels de l'enseignement secondaire en France et sont donc difficiles à exploiter par les enseignants ». Dans la mesure où la notion d'algorithme est présente dans les programmes, elle y apparaît comme une *notion outil*, une notion *non mathématisée* ou *faiblement mathématisée*, alors qu'elle pourrait apparaître « comme un véritable objet mathématique ». Cette conclusion semble quelque peu contradictoire avec le point précédent : si l'on regarde l'algorithmique comme un simple « adjuvant » de l'enseignement des mathématiques, alors on pourrait penser que sa mathématisation puisse attendre, du moins dans la mesure où cela ne handicape pas l'algorithmique dans son rôle de « fortifiant ».

3. On découvre ensuite que la « situation de recherche », manifestement articulée à un problème dit « problème des pesées », a été proposée à des *étudiants* d'université, plutôt qu'à des élèves de lycée.

4. Le résumé évoquait sur un même plan trois notions : la *construction* d'un algorithme, la *preuve* d'un algorithme et la *complexité* d'un algorithme. La conclusion revient sur cette

affirmation en détachant des deux autres notions la notion de *complexité* d'un algorithme, qui, au dire des auteurs, « reste une question difficile ». Là encore, le rôle ancillaire des concepts centraux de l'algorithmique est réaffirmé : la notion de complexité d'un algorithme, qui « pourrait trouver sa place au sein d'un enseignement de mathématiques » (c'est en fait une notion d'ores et déjà mathématisée), écrivent les auteurs, pourrait « même redonner du sens à certains des objets enseignés »…

5. Les auteurs font apparaître enfin leur travail comme participant d'un programme d'étude et de recherche plus vaste, visant à « suivre » et « accompagner » l'évolution qu'ils pensent discerner dans l'enseignement français, à savoir la « place plus grande » qu'y prendrait progressivement « l'algorithme ».

Données 1.4

☛ Introduction de l'article (pp. 51-52)

Introduction

L'algorithme est un objet méconnu. Souvent vu comme un objet de l'informatique, on l'associe à la programmation. Cependant, l'algorithme est avant tout un objet des mathématiques. « Avant tout », car la notion d'algorithme précède de beaucoup l'informatique. Le terme trouve d'ailleurs son origine dans le nom *al-Khwarizmi*, nom de l'auteur du plus ancien traité d'algèbre connu, datant du IXème siècle. Bien sûr, la formalisation de ce concept doit beaucoup aux avancées de l'informatique. Cette dernière a aussi enrichi les mathématiques de nouvelles questions. Si bien que la recherche de procédures effectives est, aujourd'hui encore plus, une problématique centrale des mathématiques. Le rôle de l'algorithme est si important qu'il est au centre d'une discipline propre, l'algorithmique, à l'intersection entre mathématique et informatique et dont l'essor actuel est considérable.

Face à cette évolution de la science mathématique et à la présence grandissante de l'informatique et de ses applications autour de nous, l'enseignement des mathématiques doit être requestionné. En 2000, la commission Kahane proposait l'introduction d'une « part d'informatique dans l'enseignement des sciences mathématiques et dans la formation des maîtres » (Kahane, 2000, p. 1). Aujourd'hui, c'est dans les programmes de seconde que l'algorithmique trouve une place. Cette présence dans l'enseignement mathématique pourrait

entraîner des malentendus entre les communautés mathématique et informatique, notamment concernant la légitimité des enseignants de mathématiques à enseigner l'algorithmique.

Et bien que ces derniers puissent penser que l'informatique est loin de leurs préoccupations, l'algorithmique relève bien de problématiques mathématiques. Se pose aussi une question de formation. A priori, peu d'enseignants ont étudié l'algorithmique en tant que telle dans leur cursus et l'algorithme risque de devenir un simple outil informatique pour les mathématiques. On peut alors s'interroger : quel accueil les enseignants de mathématiques vont-ils faire à l'algorithmique ?

Il semble donc essentiel d'aborder la question de l'introduction de l'algorithme dans l'enseignement des mathématiques d'un point de vue didactique. Ces questions ont été peu abordées et la didactique s'est surtout intéressée à la programmation et à l'outil algorithme plus qu'à l'objet mathématique. On peut citer des travaux récents tels que Nguyen (2005) qui s'intéresse à l'introduction d'éléments de programmation à l'aide de la calculatrice ou Schuster (2004) qui étudie l'introduction de problèmes d'optimisation combinatoire au secondaire. On peut aussi citer les thèses de Cartier (2008) et Ravel (2003), où l'algorithme est étudié dans des domaines mathématiques particuliers : la théorie des graphes en spécialité mathématique de terminale ES pour Cartier et l'arithmétique de la spécialité mathématique en terminale S pour Ravel. Dans cet article, c'est l'algorithme en tant qu'**objet** mathématique que nous allons étudier,

Pour mieux appréhender les questions que soulève l'algorithme, il faut se pencher sur son rôle dans les mathématiques et sur les différents aspects qu'il revêt. En s'appuyant sur la dialectique outil-objet développée par Régine Douady (1986), on peut les classer suivant qu'ils relèvent de l'aspect « outil » du concept ou de l'aspect « objet » : l'algorithme n'est pas uniquement un outil pour la résolution de problèmes mais c'est aussi un objet mathématique à part entière, pour lequel il existe un cadre d'étude précis. Précisons :

Ainsi, nous disons qu'un concept est outil lorsque nous focalisons notre intérêt sur l'usage qui en est fait pour résoudre un problème. Un même outil peut être adapté à plusieurs problèmes, plusieurs outils peuvent être adaptés à un même problème. Par objet, nous entendons l'objet culturel ayant sa place dans un édifice plus large qui est le savoir savant à un moment à un moment donné, reconnu socialement. (Douady, 1986, p. 9)

Dans une première partie, nous allons étudier plus en profondeur le concept d'algorithme afin de mieux comprendre en quoi il peut être objet d'enseignement en mathématiques. Nous nous intéresserons ensuite à la place du concept d'algorithme dans les programmes et manuels. Ce sera l'objet d'une deuxième partie. La troisième partie se centrera sur l'étude d'une situation pour manipuler l'objet algorithme.

Éléments d'analyse 1.4

1. L'introduction de l'article permet de répondre à certaines questions soulevées précédemment, notamment touchant l'usage du singulier générique « l'algorithme ». Le premier paragraphe présente cette entité – « l'algorithme » – comme un *objet*, que d'aucuns voient comme « un objet de l'informatique », mais qui, selon les auteurs, « est avant tout un objet des mathématiques ». L'usage du singulier générique côtoie ici, fugitivement, l'expression « la notion d'algorithme », « l'algorithme » étant également qualifié de « concept » lorsqu'il est versé du côté des mathématiques. Il s'agit, disent les auteurs, d'un concept dont l'impulsion viendrait historiquement des mathématiques arabes et dont la « formalisation », sinon la création, « doit beaucoup aux avancées de l'informatique ». La dernière assertion du premier paragraphe livre la clé (ou, du moins, *une* clé) de cet usage de l'expression « l'algorithme » : « Le rôle de l'algorithme est si important qu'il est au centre d'une discipline propre, l'algorithmique, à l'intersection entre mathématique et informatique et dont l'essor actuel est considérable » On notera l'emploi du substantif « mathématique », *au singulier* – *la* mathématique –, ce qui ne se produit qu'une fois dans tout l'article et semble indiquer une affirmation identitaire à son acmé : il y aurait *les* mathématiques, qui sont diverses et dont la diversification n'est jamais achevée, et *la* mathématique, en son unité épistémologique fondatrice, qui la distingue de tout le reste. Cela noté, on voit que l'usage de « l'algorithme » permet de séparer « l'algorithme », placé par l'histoire du côté des mathématiques, c'est-à-dire la notion ou le concept d'algorithme, de « l'algorithmique », champ disciplinaire hybride, en partage entre mathématiques et informatique. L'algorithmique, admettent les auteurs, ne saurait être l'affaire exclusive des mathématiques. En revanche, glissent-ils, il n'est pas niable que « la recherche de procédures effectives », c'est-à-dire la recherche… d'algorithmes, « est, aujourd'hui encore plus, une problématique centrale des mathématiques ». En conséquence, il est légitime que les mathématiques et leur enseignement « s'occupent » d'algorithmes, soit, comme disent les auteurs, « de l'algorithme ».

2. L'argumentation du premier paragraphe justifierait que l'enseignement des mathématiques inclue « l'algorithme », sinon tout à fait « l'algorithmique ». Cela justifierait aussi que l'article puisse être situé dans le champ de la didactique *des mathématiques*. Mais une conclusion aussi univoque ignorerait la tension entre deux champs : celui des mathématiques – ou de *la* mathématique – et celui de l'informatique. Le deuxième paragraphe soulève cette « difficulté » toute classique dans sa forme. Un nouvel argument est invoqué : ainsi que le soulignait la commission Kahane (2000), ce serait l'intérêt propre de l'enseignement des mathématiques que d'intégrer à son corpus « l'algorithme », comme il en va avec le nouveau programme de mathématiques de la classe de seconde en vigueur à la rentrée 2009 (voir à l'adresse http://media.education.gouv.fr/file/30/52/3/programme_mathematiques_seconde_65523.pdf). Mais cela risque d'être la source de tensions intercommunautaires, « l'informatique » et les informaticiens pouvant contester à l'enseignement des mathématiques sa légitimité – devenue pour les auteurs un devoir, semble-t-il – à enseigner « l'algorithme ». Ajoutons : surtout lorsque cette matière d'enseignement nouvelle se présente sans détour, comme dans le programme de mathématiques seconde, sous l'étiquette *Algorithmique* ! On sait que la possibilité d'une telle tension est devenue aujourd'hui structurelle avec la création, à la rentrée 2012, en classe terminale scientifique, d'un enseignement de spécialité intitulé « Informatique et sciences du numérique » (ISN), dont le programme fait apparaître quatre domaines : *Représentation de l'information*, *Algorithmique*, *Langages et programmation*, *Architectures matérielles* (http://www.education.gouv.fr/pid25535/bulletin_officiel.html?cid_bo=57572).

3. L'un des points d'ancrage de telles tensions est, très classiquement, la question de la « légitimité » – entendons plus crûment : la compétence… – des professeurs *de mathématiques* à enseigner ce qui n'est ni plus ni moins que de… l'algorithmique. À ce motif d'animadversion de la part de « l'informatique », c'est-à-dire ce motif d'hostilité *externe*, s'ajoute une *résistance interne*, lié au caractère hybride, mathématico-informatique, de « l'algorithme », qu'explicite le troisième paragraphe de l'introduction. Alors même que, selon les auteurs, « l'algorithmique relève bien de problématiques mathématiques », certains professeurs, en effet, sentant la présence de « l'informatique » derrière « l'algorithme » qu'on leur demande d'enseigner, peuvent regimber, se montrer récalcitrants, du fait d'une familiarité collective insuffisante avec la matière : « A priori », écrivent les auteurs, « peu d'enseignants ont étudié l'algorithmique en tant que telle dans leur cursus et l'algorithme risque de devenir un simple outil informatique pour les mathématiques. » On voit là encore fonctionner le couple algorithmique-algorithme comme *unissant et séparant*, d'un même mouvement, l'algorithmique (qui a partie liée avec l'informatique) et « l'algorithme », qui serait un objet

d'enseignement en *mathématiques*. Ce que doivent accepter les enseignants de mathématiques afin d'enseigner « l'algorithme », c'est… l'algorithmique. L'enseignement de « l'algorithme » suppose qu'on reconnaisse « la légitimité des enseignants de mathématiques à enseigner l'algorithmique ».

4. Le quatrième paragraphe est un lieu « critique » pour notre questionnement – l'article examiné fait-il bien connaître le fruit d'une *recherche en didactique des mathématiques* ? Pour ce qui est de l'étiquette « mathématique », nous venons de voir les auteurs la revendiquer : « l'algorithme » serait un « objet mathématique », quels que soient ses arrière-plans disciplinaires. Cela rappelé, la logique sous-tendant la première phrase du paragraphe apparaît quelque peu problématique. Les auteurs y énoncent que, étant donné l'accueil mitigé que les enseignants pourraient réserver à l'algorithmique, il leur semble « essentiel d'aborder la question de l'introduction de l'algorithme dans l'enseignement des mathématiques d'un point de vue didactique ». Peut-on comprendre qu'il ne s'agit pas, ou pas seulement, de leur proposer un exposé « purement mathématique » sur « l'algorithme », mais qu'il convient de les « choyer » *en tant qu'enseignants* en considérant les problèmes de *l'enseignement* de « l'algorithme » ? Ce souci garantirait-il que le travail des auteurs trouve place dans le champ de la didactique des mathématiques ? Oui, sans doute, si l'on entend par là une appartenance dont pourrait se réclamer tout travail sur l'enseignement et l'apprentissage de quelque matière déterminée qui explicite ses tenants et aboutissants et empiriques, et théoriques. Pourrait-il, au reste, en être autrement ? On peut imaginer un travail *mathématique* ou un travail *épistémologique*, par exemple, *pour* l'enseignement d'une matière donnée, sans qu'il s'agisse véritablement d'un travail « sur l'enseignement » de cette matière. (On peut penser ici, par exemple, à l'ouvrage de Gustave Choquet paru chez Hermann en 1964, intitulé *L'enseignement de la géométrie*, qui serait mieux nommé *Études mathématiques pour l'enseignement de la géométrie*.) Les auteurs comparent leur travail à d'autres travaux : trois sont des thèses en didactique des mathématiques (Ravel, 2003 ; Nguyen, 2005 ; Cartier, 2008), tandis qu'un quatrième (Schuster, 2004) est un article répondant au critère d'appartenance évoqué ci-dessus (http://subs.emis.de/journals/ZDM/zdm042a6.pdf). Le travail présenté aurait en tout cas un double ancrage : dans *l'enseignement* des mathématiques, d'un côté, dans les *mathématiques*, d'un autre côté, puisque, à la différence des travaux mentionnés, les auteurs s'intéressent moins à « l'outil algorithme » qu'à l'*objet mathématique* correspondant. Le premier ancrage nous promet l'explicitation par les auteurs d'un *modèle didactique de référence* (MDR) tandis que le second ancrage nous fait espérer la formulation d'un *modèle praxéologique de référence* (MPR).

5. En attendant, le cinquième paragraphe se contente d'expliciter le principe de la « dialectique outil-objet » étudié par Régine Douady (1986). À cet égard, les auteurs écrivent : « l'algorithme n'est pas uniquement un outil pour la résolution de problèmes mais c'est aussi un objet mathématique à part entière, pour lequel il existe un cadre d'étude précis » (On notera que Régine Douady a particulièrement développé la notion de « jeux de cadres », notion qui semble n'avoir guère d'écho dans la formulation précédente.) Il y a là une problématique d'étude – la dialectique entre les propriétés de l'objet et les usages de l'outil – dont il faudrait essayer de voir jusqu'à quel point elle se concrétise dans le reste de l'article.

6. Le sixième paragraphe rappelle la structure ternaire – déjà évoquée dans le résumé – donnée par les auteurs à leur présentation. Rappelons-la. La partie 1 est intitulée *L'algorithme, un objet qui questionne l'activité mathématique et l'enseignement* (pp. 53-57). Elle est décrite ici comme une étude « plus en profondeur » du « concept d'algorithme », et cela « afin de mieux comprendre en quoi il peut être objet d'enseignement en mathématiques ». La partie 2 a pour titre *L'algorithme dans l'enseignement : outil ou objet ?* (pp. 57-60). Elle est présentée ici comme décrivant « la place du concept d'algorithme dans les programmes et manuels ». La partie 3, enfin, qui reçoit le titre d'*Une situation pour l'algorithme* (pp. 60-71), étudie une « situation pour manipuler l'objet algorithme » – l'*objet* algorithme, non l'*outil* algorithme. Acceptons-en l'augure.

7. On aura noté que, dans ce dernier paragraphe, « algorithme » cède la place à « concept d'algorithme » ou à « objet algorithme ». Notons que le mot *algorithmique* apparaît 6 fois dans cette introduction, tandis que *l'algorithme* apparaît 9 fois. « Concept d'algorithme » apparaît deux fois, « notion d'algorithme » et « objet algorithme » chacun une fois. Cela confirme l'importance de la distinction « algorithme » / « algorithmique » que nous avions déjà repérée comme organisatrice du texte.

Question 2. Quel modèle praxéologique (mathématique) de référence *pour le chercheur en didactique* relatif à « l'algorithme » semble-t-il être celui des auteurs du document *E* si l'on en juge d'après la partie 1 de leur article intitulée « L'algorithme, un objet qui questionne l'activité mathématique et l'enseignement » ?

Données 2

1. – L'algorithme, un objet qui questionne l'activité mathématique et l'enseignement

1.1 *Définition de l'algorithme, différents aspects pour un même concept*

Pour présenter ces différents aspects, nous nous appuyons sur des extraits de la définition du terme « algorithme » issue du dictionnaire des mathématiques (Bouvier et *al.*, 2005). Cet ouvrage présente l'algorithme d'un point de vue généraliste et évoque les approches mathématique et informatique du concept.

Un algorithme est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données pour arriver avec certitude (c'est-à-dire sans indétermination ou ambiguïté), en un nombre fini d'étapes, à un certain résultat et cela indépendamment des données. Un algorithme ne résout donc pas un problème unique mais toute une classe de problèmes ne différant que par les données mais gouvernés par les mêmes prescriptions.

(Bouvier et *al.*, 2005, p. 27)

Cette définition, qui est assez courante dans les ouvrages généralistes, met en avant trois aspects de l'algorithme. Le premier aspect est d'être **soumis à un enjeu de vérité, de preuve** : un algorithme doit fonctionner avec *certitude* quel que soit le problème donné. Un autre aspect est que l'algorithme est un **outil de résolution de problèmes**. Plus précisément, un algorithme permet de résoudre une classe de problèmes. Le troisième aspect qui apparaît ici est qu'un algorithme est **effectif**, il s'applique à des données *finies* et résout un problème en un nombre *fini* d'étapes : il peut être mis en œuvre par un opérateur. La notion d'algorithme est indissociable de cet aspect, comme le souligne Chabert :

Aujourd'hui sous l'influence de l'informatique, la finitude devient une notion essentielle contenue dans le terme algorithme, le distinguant de mots plus vagues comme procédé, méthode ou technique. […] *Finitude du nombre des opérations et du nombre des données, mais finitude aussi de la résolution, c'est-à-dire que chaque étape doit pouvoir être réalisée selon un processus fini – ce qui n'est pas le cas, par exemple, du quotient de deux nombres réels incommensurables. On parle aussi de procédé effectif, c'est-à-dire permettant d'obtenir effectivement le résultat (en un temps fini).*

(Chabert, 1994, p. 6)

Une autre question se pose alors : Tout ce qui est effectif, une formule par exemple, est-il un algorithme ?

Pour Bouvier et *al.* (2005) la réponse est oui. Plus précisément, une formule peut être vue comme un algorithme dont les prescriptions ne varient pas en fonction des données. Il semble donc que la notion d'algorithme englobe celle de procédure effective, mais nous verrons plus loin que ces « algorithmes-formules » ont un intérêt limité du point de vue de l'algorithmique (paragraphe 2.2).

Le développement de l'informatique a bien sûr une influence importante sur l'algorithmique et soulève des questions d'implémentation de l'algorithme. Ces questions de programmation et de langages informatiques ne seront pas étudiées ici. Comme le souligne Bouvier et *al.* (2005), des questions se posent aussi quant à l'efficacité de la mise en œuvre de ces algorithmes : la question des temps de calcul, c'est-à-dire le nombre d'étapes élémentaires qu'effectue un algorithme en fonction de la taille des données, mais aussi la question de la taille mémoire, c'est-à-dire la place nécessaire pour stocker les différentes structures de données, correspondant à la complexité en temps et en espace de l'algorithme.

Un nouvel aspect est ici évoqué : **la complexité**.

Les questions d'existence d'algorithme pour certains problèmes et de leur complexité ont conduit à un cadre théorique d'étude de l'algorithme en tant qu'objet : l'hypothèse fondamentale de la théorie des algorithmes est que tout algorithme peut être réalisé par une machine de Turing particulière et aussi, par conséquent par une machine de Turing universelle [1 Une machine de Turing est un modèle abstrait du fonctionnement des appareils mécaniques de calcul, créé par Alan Turing afin de donner une définition précise du concept d'algorithme. Pour plus de détails on pourra consulter : Pierre Wolper. Introduction à la calculabilité, cours et exercices corrigés, 3ème éd. Dunod (2006).] Cela constitue un autre aspect de l'algorithme.

Un dernier aspect de l'algorithme peut être soulevé, l'aspect **constructif** de l'algorithme : En ce qui concerne les problèmes d'existence, qui divisent les mathématiciens sur le plan philosophique, les algorithmes permettent d'apporter une méthode de construction de l'objet reconnue par l'ensemble des mathématiciens (Bouvier et *al.*, 2005).

Parmi les différents aspects relevés certains font référence à l'outil, d'autres à l'objet. Regarder l'algorithme en tant qu'**objet**, c'est s'intéresser aux questions de bon fonctionnement, de domaine de validité, de complexité et de description des algorithmes. Ce sont des problématiques de l'**algorithmique**. Les aspects preuve, complexité, machine de Turing et constructif réfèrent à l'objet algorithme.

Regarder l'algorithme en tant qu'**outil**, c'est s'intéresser à l'utilisation que l'on en fait pour résoudre des problèmes. Les aspects résolution de problème, effectivité et formule réfèrent à l'outil algorithme.

Ce sont donc les questions de construction, de preuve et de complexité des algorithmes qui font vivre l'algorithme en tant qu'objet des mathématiques et c'est autour d'elles que notre étude s'articule.

1.2 *L'algorithmique fait-elle appel à un mode de pensée spécifique ?*

Knuth (1985), qui exerce indépendamment des activités mathématiques et informatiques, s'interroge sur les différences qui existent entre la *pensée mathématique* et la *pensée algorithmique* (l'algorithmique étant, pour lui, entendue au sens large, comme la science informatique). Il repère dans l'activité du mathématicien neuf grands *modes de pensée* et remarque que six d'entre eux sont communs à la pensée algorithmique :

– la manipulation de formules
– la représentation d'une réalité
– la réduction à des problèmes plus simples
– le raisonnement abstrait
– les structures d'informations
– les algorithmes.

Il ajoute deux catégories à la pensée algorithmique qui lui semblent ne pas être présentes dans la pensée mathématique [2 Knuth souligne tout de même que certains textes mathématiques sont extrêmement proches de ce qu'il appelle la pensée algorithmique, en particulier un livre d'analyse constructive : Bishop E. Foundations of constructive Analysis. McGraw Hill (1967).] :

– la notion de complexité
– la notion d'affectation symbolisée par := ou ←. Précisons ici que Knuth fait référence à une notion d'affectation dynamique : l'affectation successive de différentes valeurs à une même variable au cours des différentes étapes d'un processus.

La notion de complexité semble donc propre à l'algorithmique. Knuth fait remarquer que, dans les résultats constructifs rencontrés en mathématiques, le plus souvent, on ne tient pas compte du « coût » de la construction, pour lui tous les ingrédients d'un algorithme ne sont pas rassemblés.

La notion d'affectation n'est pas sans rappeler la distinction entre *variable mathématique* et *variable informatique*. Une variable mathématique est un symbole représentant un élément non

spécifié ou inconnu d'un ensemble ou jouant un rôle de marque place (en logique par exemple). Une variable informatique désigne un emplacement dans la mémoire, son contenu peut changer. L'opération d'affectation (relation antisymétrique) diffère de la relation d'égalité (évidemment symétrique).

L'algorithmique fait appel à des raisonnements communs aux mathématiques, mais aussi à un mode de pensée spécifique. Il semble essentiel que l'étude de l'algorithme en tant qu'objet permette la confrontation à ce mode de pensée. Une préoccupation centrale des mathématiques est de fournir des démonstrations de ses résultats, il est donc inévitable, si l'on regarde l'algorithme comme un objet des mathématiques, d'interroger le lien algorithme-preuve.

1.3 *Algorithme et preuve*

L'algorithme entretient un lien étroit avec la preuve, de diverses manières. Pour mieux les comprendre commençons par regarder un exemple :

Algorithme de recherche du pgcd

L'existence du *pgcd* de deux entiers peut être prouvée comme suit :

Preuve : Soient a et b deux entiers avec $a \geq b$. Notons D l'ensemble des diviseurs communs à a et b. Comme $1 \mid a$ et $1 \mid b$, on a $D \neq \emptyset$, et D est inclus dans $\{1, \ldots, a\}$. D est borné non vide, donc possède un plus grand élément d noté $pgcd(a, b)$.

Le *pgcd* peut être obtenu par l'algorithme d'Euclide [3 On utilise ici quelque notations propres à l'informatique et à la logique pour simplifier la présentation.] :

Euclide(a, b) :
$r_1 \leftarrow a$
$r_2 \leftarrow b$
tant que $r_2 \neq 0$
$r_1 = q.r_2 + r$ *(division euclidienne)*
$r_1 \leftarrow r_2$
$r_2 \leftarrow r$
retourner r_1

Il n'est pas évident que cet algorithme fonctionne et construise le *pgcd* de a et b. Il est nécessaire de le prouver :

Preuve :

– Preuve de la correction [4 c'est-à-dire : preuve que l'algorithme donne bien le *pgcd*.]. Notons *ak, bk* et *qk* les valeurs successives prises par r_1, r_2 et q après k pas de la boucle *tant que*. On a pour tout k : $a_k = q_k \, . \, b_k + b_{k+1}$. Donc pour tout entier d, ($d \mid a_k$ et $d \mid b_k$) ssi ($d \mid a_{k+1}$ et $d \mid b_{k+1}$). D'où $pgcd(a_k, b_k) = pgcd(a_{k+1}, b_{k+1})$.
Ainsi lorsque l'algorithme s'arrête après n pas : $pgcd(a, b) = pgcd(a_n, 0) = a_n$.
– Preuve de terminaison [5 c'est-à-dire : preuve donne le résultat après un nombre fini d'étapes.]. On remarque que la suite (b_k) est strictement décroissante. Donc la condition d'arrêt $b_k = 0$ sera vérifiée après un nombre fini d'itérations et l'algorithme s'arrêtera.

Revenons maintenant sur les liens qu'entretiennent algorithme et preuve.

Preuve d'algorithme

Tout d'abord, pour présenter un algorithme et affirmer qu'il résout une famille de problèmes, il convient d'en donner une démonstration. Il faut être certain qu'il aboutit au bon résultat quel que soit le problème en instance, c'est ce que l'on appelle la **correction**, et pouvoir garantir que ce résultat sera atteint en un nombre fini d'étapes, c'est la **terminaison**. C'est ce que nous avons illustré dans l'exemple précédent.

De plus un algorithme peut être utilisé au cours d'une preuve en tant qu'inférence (par exemple lorsque l'on fait appel à l'algorithme du *pgcd* pour prouver l'existence des coefficients de Bezout). La preuve de l'algorithme donné est alors un point indispensable à la validation de la démonstration concernée.

La preuve d'algorithmes revêt aussi une importance particulière depuis l'utilisation d'algorithme dans des preuves récentes, en tant que substitut au mathématicien pour des tâches non accessibles à l'homme (théorème des quatre couleurs par exemple).

L'algorithme dans l'activité mathématique

Les preuves dites constructives (la preuve par induction en est un exemple) peuvent en général donner lieu à un algorithme. Il peut être nécessaire d'expliciter les algorithmes sous-jacents. Ce type de preuves constructivistes concerne le plus souvent des problèmes d'existence d'objets ou de reconnaissance d'une classe d'objets.

Afin de démontrer l'existence d'un objet ayant une certaine propriété *(P)* les méthodes constructives exhibent de tels objets. En mathématiques, la question de la caractérisation de tous les objets vérifiant *(P)* s'avère une question de recherche souvent difficile. Un moyen de répondre à cette question est l'énumération de tous ces objets : cette technique peut aussi faire appel à un algorithme. Cependant le problème d'énumération est parfois encore trop difficile.

De plus, l'algorithmique permet d'aborder des questions de reconnaissance des objets vérifiant *(P)* : étant donné un objet *O*, un algorithme de reconnaissance permet de dire si *O* vérifie *(P)*. Ces questions sont clairement de nature mathématique et ne peuvent être abordées sans le concept d'algorithme.

Reprenons l'exemple des preuves d'existence. Pour un même résultat, on peut distinguer deux types de preuves :

– Les preuves d'existence « théoriques », sans construction explicite de l'objet : c'est souvent le cas des preuves par l'absurde ou des preuves probabilistes. La preuve d'existence du *pgcd* donnée plus haut en est un exemple.
– Les preuves constructives parmi lesquels il faut distinguer les preuves algorithmiques qui donnent une méthode de construction des objets et la construction explicite des objets considérés.

Le plus satisfaisant est sans doute de bénéficier d'une construction explicite et d'un algorithme de reconnaissance de la classe d'objets. On dira alors que le problème de reconnaissance relatif au théorème d'existence est entièrement résolu.

Complexité, optimalité et preuve

L'étude de la complexité d'algorithmes est une autre problématique en lien avec l'activité de preuve. La complexité peut être abordée sous divers angles dont deux sont classiques dans l'étude d'algorithmes : la *complexité au pire* et la *complexité en moyenne* [6 Ces deux notions de complexité seront définies précisément à la fin du paragraphe 3.1.] Se pose alors la question de recherche d'algorithmes optimaux pour un problème donné. Là encore, la preuve joue un rôle fondamental. L'apprentissage de la preuve est une problématique centrale de l'enseignement des mathématiques. Au vu des liens que l'algorithme entretient avec la preuve, il est naturel de se poser la question suivante : *L'algorithme peut-il être un levier efficace pour l'apprentissage de la preuve ?* (pp. 53-57)

Éléments d'analyse 2

1. Les auteurs se réfèrent d'abord à une « définition » de la notion d'algorithme donnée dans un ouvrage généraliste (non spécialisé), le *Dictionnaire des mathématiques* d'Alain Bouvier, Michel George et François Le Lionnais (ouvrage qui, en 2013, en était à sa neuvième édition, la première datant de 1979). Sans doute s'agit-il là d'une référence regardée par ces auteurs comme convenant à la revue dans laquelle leur article doit être publié. De cette définition, ils

dégagent trois « aspects » de la notion d'algorithme : un algorithme doit résoudre toute une classe de problèmes (par exemple trouver le PGCD de deux entiers), il doit conduire au résultat attendu (on obtient bien le PGCD), et il doit faire cela de manière « effective », en un nombre fini de pas. En vérité ce dernier point notamment est quelque peu *flou* – par exemple, « fini » ne signifie par « borné » (un entier peut être « aussi grand qu'on veut »). Il est vrai que les auteurs introduiront, plus loin dans cette section, la notion de *terminaison* – d'arrêt –, introduite par Alan Turing dans son article fondateur de 1936, « On Computable Numbers, with an Application to the *Entscheidungsproblem* ». Mais ce que ne disent pas nettement les auteurs de l'article étudié, c'est que la notion d'algorithme ne possède pas, aujourd'hui, de définition simple et claire, univoque et universellement admise. Par contraste, voici, à titre d'exemple, un extrait d'un document en ligne intitulé « Qu'est-ce qu'un algorithme ? » (http://www.scriptol.fr/programmation/algorithme-definition.php), où référence est faite à l'un des plus célèbres algorithmiciens, Donald Ervin Knuth (prononciation : /kəˈnuːθ/) :

> Knuth (1968, 1973) a donné une liste de cinq propriétés qui sont largement reconnues comme les pré-requis d'un algorithme :
>
> 1. **Finitude** : « Un algorithme doit toujours se terminer après un nombre fini d'étapes. »
> 2. **Définition précise** : « Chaque étape d'un algorithme doit être définie précisément; les actions à transposer doivent être spécifiées rigoureusement et sans ambiguïté pour chaque cas. »
> 3. **Entrées** : « ... des quantités qui lui sont données avant qu'un algorithme ne commence. Ces entrées sont prises dans un ensemble d'objets spécifié. »
> 4. **Sorties** : « ... des quantités qui ont une relation spécifiée avec les entrées. »
> 5. **Rendement** : « ... toutes les opérations que l'algorithme doit accomplir, doivent être suffisamment basiques pour pouvoir être en principe réalisées dans une durée finie par un homme utilisant du papier et un crayon. »
>
> Knuth offre comme exemple l'algorithme d'Euclide pour déterminer le plus grand commun diviseur [de] deux nombres entiers naturels.
>
> Knuth admet que, même si sa description de ce qu'est un algorithme peut être intuitivement claire, il lui manque la rigueur formelle, tandis que n'est pas exactement clair ce que *précisément défini* veut dire, ou *spécifié rigoureusement et sans ambiguïté* signifie, ou *suffisamment basique*, et ainsi de suite.

On se trouve en vérité dans une situation historique dans laquelle l'objet principal de l'étude, la notion d'algorithme, *n'est pas entièrement mathématisé*, même s'il l'est partiellement, fragmentairement. En d'autres termes, la matière à étudier est en même temps une matière *en voie de mathématisation*, dont certains aspects sont d'ores et déjà mathématisés, dont d'autres

aspects peuvent être jugés mathématisables, tandis que d'autres encore semblent échapper, pour un temps non précisé, à un tel processus. À cet égard, on peut dire, avec les auteurs, que la notion d'algorithme « questionne l'activité mathématique et l'enseignement [des mathématiques] », en ce sens qu'elle appelle un *enseignement de la mathématisation* (mot qui n'apparaît pas dans l'article examiné), qu'on doit regarder comme se situant au cœur de tout enseignement des mathématiques. Par rapport à cette situation, la question brutale « La notion d'algorithme est-elle une notion mathématique ? » doit donc être prise *cum grano salis* – avec quelque recul.

2. Les auteurs écartent de leur champ de travail ce qu'ils nomment les « algorithmes-formules », dans lesquels « les prescriptions ne varient pas en fonction des données » (on peut songer à la résolution des équations du second degré, par exemple). De tels algorithmes se révèlent peu intéressants « du point de vue de l'algorithmique », affirment-ils. C'est là sans doute une assertion qu'il est loisible d'interroger, en particulier *du point de vue didactique*, même si un algorithme-formule est un objet mathématiquement (et algorithmiquement) « simple ». Notons ici que les auteurs confirmeront ce parti-pris plus loin dans leur article.

3. Il faut distinguer entre ce qui serait la *définition* de la notion d'algorithme et ce que seraient les *propriétés* éventuelles d'un algorithme. Le fait de se terminer pour toutes les valeurs des données considérées est-il une partie intégrante de la définition de la notion d'algorithme ? Ou bien est-ce une propriété éventuelle d'un algorithme ? Il en va de même du fait de donner en sortie la solution au problème correspondant aux données proposées en entrée, c'est-à-dire de la propriété de « correction » de l'algorithme, qui appelle une *preuve* de l'algorithme.

4. D'une façon plus générale, la section de l'article examinée ici condense dans un espace limité *un grand nombre de considérations*, ce que donne à voir le découpage de cette section, découpage que nous reproduisons ici :

1. – L'algorithme, un objet qui questionne l'activité mathématique et l'enseignement

1.1 *Définition de l'algorithme, différents aspects pour un même concept*

1.2 *L'algorithmique fait-elle appel à un mode de pensée spécifique ?*

1.3 *Algorithme et preuve*

Algorithme de recherche du pgcd

Preuve d'algorithme

L'algorithme dans l'activité mathématique

On se trouve là devant *une difficulté classique dans les articles de recherche en didactique* (et pas seulement en didactique des mathématiques) : les auteurs semblent avoir hésité entre d'une part le fait de faire connaître au lecteur leur *modèle praxéologique de référence* – qui renvoie *en partie* à des « savoirs savants » – et d'autre part le fait de supposer les constituants de ce modèle « connus », voire « bien connus » – même quand, à l'évidence, il n'en est rien. Par contraste, un certain nombre de développements *précis* pourraient être attendus dans un article de recherche, et cela, éventuellement, sous forme d'*annexes* à l'article, afin d'expliciter plus nettement (et plus complètement) les notions invoquées.

① Pour le contraste, considérons l'extrait ci-après d'un texte « spécialisé », le Cours INF423 de l'École Polytechnique, dû à Olivier Bournez et intitulé *Fondements de l'informatique. Logique, modèles, et calculs* (version de septembre 2013, qu'on trouvera en ligne à l'adresse suivante : http://www.enseignement.polytechnique.fr/informatique/INF423/uploads/Main/poly-good.pdf) :

10.1 Complexité d'un algorithme

On considère donc typiquement dans ce chapitre un problème $\mathcal{P}$ pour lequel on connaît un algorithme $\mathcal{A}$: cet algorithme, on sait qu'il est correct, et qu'il termine. Il prend en entrée une donnée d, et produit un résultat en sortie $\mathcal{A}(d)$ en utilisant certaines ressources (on ne parle donc plus que de problèmes décidables).

Exemple 10.1 Le problème $\mathcal{P}$ peut par exemple être celui de déterminer si un nombre v est parmi une liste de n nombres.

Il est clair que l'on peut bien construire un algorithme $\mathcal{A}$ pour résoudre ce problème : par exemple,

– on utilise une variable res que l'on met à 0 ;

– on parcourt la liste, et pour chaque élément :

– on regarde si cet élément est le nombre v :

– si c'est le cas, alors on met la variable res à 1

– à l'issue de cette boucle, on retourne res.

Cet algorithme n'est pas le plus efficace que l'on puisse envisager. D'une part, on pourrait s'arrêter dès que l'on a mis res à 1, puisqu'on connaît la réponse. D'autre part, on peut clairement faire tout autrement, et utiliser par exemple une recherche par dichotomie (un algorithme récursif), si l'on sait que la liste est triée.

10.1.1 Premières considérations

On mesure toujours l'efficacité, c'est-à-dire la complexité, d'un algorithme en termes d'une mesure élémentaire μ à valeur entière : cela peut être le nombre d'instructions effectuées, la taille de la mémoire utilisée, ou le nombre de comparaisons effectuées, ou toute autre mesure.

Il faut simplement qu'étant donnée [*sic*] une entrée d, on sache clairement associer à l'algorithme $\mathcal{A}$ sur l'entrée d, la valeur de cette mesure, notée $\mu(\mathcal{A}, d)$: par exemple, pour un algorithme de tri qui fonctionne avec des comparaisons, si la mesure élémentaire μ est le nombre de comparaisons effectuées, $\mu(\mathcal{A}, d)$ est le nombre de comparaisons effectuées sur l'entrée d (une liste d'entiers) par l'algorithme de tri $\mathcal{A}$ pour produire le résultat $\mathcal{A}(d)$ (cette liste d'entiers triée).

La fonction $\mu(\mathcal{A}, d)$ dépend de $\mathcal{A}$, mais aussi de l'entrée d. La qualité d'un algorithme $\mathcal{A}$ n'est donc pas un critère absolu, mais une fonction quantitative $\mu(\mathcal{A}, .)$ des données d'entrée vers les entiers.

10.1.2 Complexité d'un algorithme au pire cas

En pratique, pour pouvoir appréhender cette fonction, on cherche souvent à évaluer cette complexité pour les entrées d'une certaine *taille* : il y a souvent une fonction *taille* qui associe à chaque donnée d'entrée d, un entier $taille(d)$, qui correspond à un paramètre naturel. Par exemple, cette fonction peut être celle qui compte le nombre d'éléments dans la liste pour un algorithme de tri, la taille d'une matrice pour le calcul du déterminant, la somme des longueurs des listes pour un algorithme de concaténation.

Pour passer d'une fonction des données vers les entiers, à une fonction des entiers (les tailles) vers les entiers, on considère alors la complexité *au pire cas* : la complexité $\mu(\mathcal{A}, n)$ de l'algorithme $\mathcal{A}$ sur les entrées de taille n est définie par

$$\mu(\mathcal{A}, n) = \max_{d \text{ entrée avec } taille(d) = n} \mu(\mathcal{A}, d).$$

Autrement dit, la complexité $\mu(\mathcal{A}, n)$ est la complexité la pire sur les données de taille n.

Par défaut, lorsqu'on parle de complexité d'algorithme en informatique, il s'agit de complexité au pire cas, comme ci-dessus.

Si l'on ne sait pas plus sur les données, on ne peut guère faire plus que d'avoir cette vision pessimiste des choses : cela revient à évaluer la complexité dans le pire des cas (le meilleur des cas n'a pas souvent un sens profond, et dans ce contexte le pessimisme est de loin plus significatif).

10.1.3 Complexité moyenne d'un algorithme

Pour pouvoir en dire plus, il faut en savoir plus sur les données. Par exemple, qu'elles sont distribuées selon une certaine loi de probabilité.

Dans ce cas… (pp. 156-157)

Il s'agit là d'un extrait du chapitre 10, intitulé « Bases de l'analyse de complexité d'algorithmes », qui vient après plus de 150 pages du cours mentionné. On ne s'étonnera donc pas qu'il ne soit pas totalement *self-contained* et sollicite certaines connaissances « antécédentes ». Mais ce devrait être le rôle de l'exposé du modèle praxéologique de référence des chercheurs que de proposer une présentation dépendant le moins possible de connaissances allogènes.

② On a rappelé que les auteurs de l'article examiné évoquent la notion d'*optimalité*. Voici le passage du cours d'Olivier Bournez sur cette notion :

10.2 Complexité d'un problème

On peut aussi parler de la *complexité d'un problème* : cela permet de discuter de l'optimalité ou non d'un algorithme pour résoudre un problème donné.

On fixe un problème $\mathcal{P}$: par exemple celui de trier une liste d'entiers. Un algorithme $\mathcal{A}$ qui résout ce problème est un algorithme qui répond à la spécification du problème $\mathcal{P}$: pour chaque donnée d, il produit la réponse correcte $\mathcal{A}(d)$.

La complexité du problème $\mathcal{P}$ sur les entrées de taille n est définie par

$$\mu(\mathcal{P}, n) = \inf_{\mathcal{A} \text{ algorithme qui résout } \mathcal{P}} \max_{d \text{ entrée avec } taille(d) = n} \mu(\mathcal{A}, d).$$

Autrement dit, on ne fait plus seulement varier les entrées de taille n, mais aussi l'algorithme. On considère le meilleur algorithme qui résout le problème. Le meilleur étant celui avec la meilleure complexité au sens de la définition précédente, et donc au pire cas. C'est donc la complexité du meilleur algorithme au pire cas.

L'intérêt de cette définition est le suivant : si un algorithme $\mathcal{A}$ possède la complexité $\mu(\mathcal{P}, n)$, c'est-à-dire est tel que $\mu(\mathcal{A}, n) = \mu(\mathcal{P}, n)$ pour tout n, alors cet algorithme est clairement optimal. Tout autre algorithme est moins performant, par définition. Cela permet donc de prouver qu'un algorithme est optimal.

③ Voici maintenant – à nouveau – le passage de l'article examiné intitulé *Complexité, optimalité et preuve* :

L'étude de la complexité d'algorithmes est une autre problématique en lien avec l'activité de preuve. La complexité peut être abordée sous divers angles dont deux sont classiques dans l'étude d'algorithmes : la *complexité au pire* et la *complexité en moyenne* [6 Ces deux notions de complexité seront définies précisément à la fin du paragraphe 3.1.] Se pose alors la question

de recherche d'algorithmes optimaux pour un problème donné. Là encore, la preuve joue un rôle fondamental. L'apprentissage de la preuve est une problématique centrale de l'enseignement des mathématiques. Au vu des liens que l'algorithme entretient avec la preuve, il est naturel de se poser la question suivante : *L'algorithme peut-il être un levier efficace pour l'apprentissage de la preuve ?*

Quel sens donner, ici, à l'affirmation selon laquelle « la preuve joue un rôle fondamental » dès lors qu'on se pose la question de la recherche « d'algorithmes optimaux pour un problème donné » ? S'agit-il simplement de rappeler que, si l'on recherche l'algorithme optimal résolvant un problème donné parmi un certain ensemble d'algorithmes, il convient de s'assurer que ces algorithmes résolvent bien ledit problème ? C'est ce qu'on peut penser en relisant le passage suivant :

Tout d'abord, pour présenter un algorithme et affirmer qu'il résout une famille de problèmes, il convient d'en donner une démonstration. Il faut être certain qu'il aboutit au bon résultat quel que soit le problème en instance, c'est ce que l'on appelle la **correction**, et pouvoir garantir que ce résultat sera atteint en un nombre fini d'étapes, c'est la **terminaison**. C'est ce que nous avons illustré dans l'exemple précédent. (p. 56)

« L'exemple précédent » auquel se réfèrent les auteurs est, nous le savons, celui du calcul du PGCD de deux entiers.

5. L'exposé de la sous-section 1.3, intitulée « Algorithme et preuve », n'introduit pas la notion d'*invariant de boucle*, qui est pourtant – de façon implicite – au cœur de la démonstration proposée. Pour le contraste, à nouveau, mais aussi pour enrichir notre propre modèle praxéologique de référence (nécessaire pour analyser des recherches dans le domaine de la didactique de l'algorithmique), suivons maintenant un exposé sur cette notion.

① Cette présentation est empruntée au livre classique de John D. Lipson, *Elements of Algebra and Algebraic Computing* (1981, Reading, Mass. : Addison-Wesley). Commençons par une convention de notation :

Assertions about program variables. Let σ be a program statement (command) and let p and q be assertions about the values of variables manipulated by the program in which σ appears. We write

$$\{p\}\ \sigma\ \{q\}$$

to mean: if p holds before execution of σ, then q holds after execution of σ. In this context, we refer to p as the *pre-condition* of σ and to q as the *post-condition* of σ. (p. 218)

② Voici alors le théorème clé :

The Invariant Relation Theorem. In a **while** loop

while γ **do** σ

we assume that σ cannot cause a premature transfer from the loop; the loop terminates only by having γ become false. Under this stipulation, we can establish the following theorem, the main result of this appendix, which describes the semantics of a **while** loop in terms of assertions (about program variables):

Theorem *(Invariant Relation Theorem).* Let ℓ be a **while** loop

ℓ: **while** γ **do** σ

and let *p* be a program assertion. If

1. (a) *p* holds on entry to ℓ,

   (b) {*p* and γ} σ {*p*}

2. the loop ℓ terminates,

then *p* and ¬ γ hold on exit from ℓ (¬ γ means "not γ").

Proof of the IR Theorem. By hypothesis 2 the loop terminates, say after *n* iterations for some $n \geq 0$. Since termination can only occur by having γ false, we have the "¬ γ" part of the desired conclusion *p* and ¬ γ. We now establish the "*p*" part by showing the *invariance* of *p*: that *p* holds after *k* iterations for $k = 0, 1, \ldots, n$. The proof is by induction on *k*.

Basis ($k = 0$). *p* holds by hypothesis 1(a).

Induction. Assume that *p* holds after *k* iterations for $0 \leq k < n$. Since $k < n$, γ must be true at the beginning of the (k + l)st iteration. By hypothesis 1(b), *p* holds at the end of the (k + 1)st iteration, which completes the proof of the invariance of *p* and completes the proof of the IR theorem.□

Remarks. 1. The assertion *p* in the IR theorem is called an *invariant relation,* or *invariant*, of the given loop, as characterized by hypotheses 1(a) and 1(b). Hence the name *Invariant Relation* Theorem.

2. Some readers might prefer to regard the IR Theorem as an *axiom*, one which *defines* the semantics of the **while** statement. Fine. (pp. 219-220)

③ Voici maintenant un exemple typique d'utilisation du théorème de l'invariant de boucle, ici pour donner une preuve de l'algorithme, c'est-à-dire une démonstration du fait que l'algorithme est *correct*.

Example 1. Let us verify that the program of Fig. 3, for integers $a \geq 0$ and $b > 0$, correctly computes *q* and *r* satisfying the Division Property: $a = bq + r$, $0 \leq r < b$ (*).

```
        begin {a ≥ 0, b > 0}
1.          q := 0;
2.          r := a;
            {p: a = bq + r and r ≥ 0}
3.      ℓ: while r ≥ b do
            begin
4.              r := r – b;
5.              q: = q + 1
            end
        end
```

Fig. 3 Division program for **N**.

The comments enclosed by braces { } are assertions about program variables. The assertion preceding line 1 is our assumption about the input integers a and b. The assertion preceding the **while** loop ℓ of line 3 is the crucial one. The claim is that the assertion there,

$$p : a = bq + r \text{ and } r \geq 0,$$

is a loop invariant of ℓ.

But before proving the invariance of p, let us jump to the conclusion p and $\neg\, \gamma$ of the IR Theorem. Here the loop condition γ is "$r \geq b$," so that

$$p \text{ and } \neg\, \gamma \Rightarrow a = bq + r \text{ and } r \geq 0 \text{ and } r < b$$

$$\Rightarrow a = bq + r \text{ and } 0 \leq r < b.$$

Thus the conclusion of the IR Theorem is precisely the program correctness verification that we are after: after the loop terminates, q and r satisfy the Division Property (*).

To complete the verification, we must of course check out the two hypotheses of the IR Theorem.

1. (*Invariance of p.*)

(a) On entry to ℓ, $q = 0$ and $r = a \geq 0$ by lines 1 and 2, so that p holds trivially.

(b) ($\{p$ and $\gamma\}\ \sigma\ \{p\}$.) Assume p and γ before execution of the loop statement σ (lines 4 and 5). We denote the redefined values of r and q after execution of σ by r' and q'. (We must show that p holds for these primed values.)

According to σ we have

$$r' = r - b,\ q' = q + 1,$$

so that

$$bq' + r' = b(q + 1) + (r - b)$$

$$= bq + r$$

$$= a,$$

the last equality following because by assumption p holds before execution of σ. Also, $r \geq b$ by γ, so that $r' = r - b \geq 0$. Hence p holds after execution of σ. p is thus a loop invariant as claimed.

2. (*Termination.*) Successive values of r, one value per iteration of the loop ℓ, constitute a strictly decreasing sequence of nonnegative integers $\leq a$ (*strictly* decreasing by line 4 and $b > 0$). Any such sequence must be finite, proving termination.

We have just shown that the two hypotheses of the IR Theorem hold, hence the conclusion (p and $\neg \gamma$) holds, which, as we have already seen, lets us conclude the correctness of our division program. (pp. 220-221)

6. L'auteur que nous venons de suivre ajoute alors (c'est nous qui soulignons) : « We now change the viewpoint from program *verification* to program *construction* (more specifically, loop derivation). It is in this realm that the IR Theorem finds its most rewarding applications. »

① Voici d'abord la problématique générale de l'auteur :

In broad outline, the overall strategy is to find a relation p and a condition γ such that p and $\neg \gamma$ together yield a desired computational state of affairs. The IR Theorem tells us that p and $\neg \gamma$ will be achieved by executing the following high level program segment:

initialize variables so that p holds;
while γ **do**
σ: maintain p as an invariant
and do measurable work.

The initialization statement assures that hypothesis 1(a) of the IR Theorem is satisfied; the specification of the loop statement σ assures that hypotheses 1(b) and 2 are satisfied. In particular, what is meant by the "measurable work" part of σ is this: execution of σ brings the loop closer to termination—one step closer!—and termination is assured after finitely many iterations of σ. Since the hypotheses of the IR Theorem are satisfied, we can conclude p and $\neg \gamma$ on exit from the **while** loop, as already claimed.

② L'auteur met en œuvre la technique de construction indiquée à propos de l'exemple suivant.

Example 1. Devise a program to compute

$$y = x^n \ (n \geq 0)$$

in a multiplicative monoid. (A simple problem to be sure, one for which we do not need the help of any high-powered mental aids. However, let us derive a program via the IR Theorem—we

promise a payoff in the next example.)

The sequence

$$x^n = x^{n-1}x = x^{n-2}x^2 = \ldots = x^0x^n$$

suggests that we manipulate program valuables [*sic*] u and v so that

$$p\text{: } x^n = x^u v \text{ and } u \geq 0$$

is maintained as an invariant of a loop "**while** γ **do** σ." Our task now is to derive the loop, including its initialization.

Loop condition γ. On termination we want $u = 0$, since the above loop invariant p then gives $x^n = v$; i.e., v yields the desired computational result. Therefore we take the loop condition γ to be "$u \neq 0$."

Initialization. If $u = n$ and $v = 1$ initially, then p holds trivially on entry to the loop.

Our partially refined program is then as given in Fig. 4.

---

begin $\{n \geq 0\}$
    $u := n$;
    $v := 1$;
    $\{p\text{: } x^n = x^u v \text{ and } u \geq 0\}$
    while $u \neq 0$ **do**
    σ: redefine u, v by u', v' so that p is maintained and measurable work is done
end
$\{x^n = v\}$

Fig.4 Partially refined program for computing x^n.

Now for the refinement of the loop statement σ. Before execution of σ we can assume p and γ, i.e., $x^n = x^u v$ and $u > 0$. Therefore we have

$$x^n = x^u v = x^{u-1}(xv) \text{ [with } u' = u-1,\ v' = xv\text{]}$$

As indicated, p will be maintained by σ if σ redefines u and v by $u' = u - 1$ and $v' = x \times v$. (Note in particular that $u > 0$ implies $u' \geq 0$.) Also, σ performs measurable work: $u' = u - 1$ means that the loop terminates after n iterations. This completes the derivation of the program of Fig. 5, *correct by construction*, for computing x^n.

---

begin $\{n \geq 0\}$
        $u := n$;
        $v := 1$;
        $\{p\text{: } x^n = x^u v \text{ and } u \geq 0\}$
        while $u \neq 0$ **do**
        begin
                $u := u - 1$;
                $v := x \times v$
        end
end
$\{x^n = v\}$

Fig. 5 Simple program for computing x^n.

We have called the program of Fig. 5 "simple," and indeed it is. However, this simple program requires n iterations to compute x^n, which can be prohibitively costly for *very* large values of n, say in the range $10^5 - 10^8$. (…) Our goal, therefore is to speed up our simple program. (pp. 222-224)

③ Dans une nouvelle étape, l'auteur montre donc comment « améliorer » l'algorithme « simple » précédemment obtenu.

Example 2 (*Fast computation of* x^n), We retum to the partially refined program of Fig. 4. Our task is clear: to increase the "measurable work" carried out by each iteration of σ.

In our simple program of Fig. 5 we forced the so-called induction variable u to zero by exploiting

$$u = (u - 1) + 1$$

(which led to $x^u v = x^{(u-1)+1} v = x^{u'} v'$ where $u' = u - 1$, $v' = xv$). Rather than subtract one from u, we divide u by two which gives by the Division Property

$$u = 2(u \text{ div } 2) + (u \text{ mod } 2).$$

We now generalize the invariant relation p of Example 1 by the introduction of a variable t:

$$p : x^n = t^u v,\ u \geq 0.$$

If $u > 0$, we have

$$\begin{aligned} x^n &= t^u v \\ &= \{t^{[2(u \text{ div } 2) + (u \text{ mod } 2)]}\} v \\ &= (t^2)^{u \text{ div } 2} (t^{u \text{ mod } 2} v) \text{ [with } t' = t^2,\ u' = u \text{ div } 2,\ v' = t^{u \text{ mod } 2} v] \end{aligned}$$

We see that the loop statement σ of Fig. 4 will preserve our new invariant p if σ redefines t, u, v by

$$\begin{aligned} t' &= t^2, \\ u' &= u \text{ div } 2, \\ v' &= v \text{ if } u \text{ is even} \\ &= tv \text{ otherwise.} \end{aligned}$$

These considerations lead to the program of Fig. 6, where we have appropriately initialized t. (Note that the new values t', u', v' of t, u, v are defined in terms of the old values. For this reason, in the while loop of Fig. 6 it is crucial that the redefinition of v precede the redefinitions of t and u.)

We have called the program of Fig. 6 "fast." Let us see how fast. The program computes a

sequence

$$u_0 = n,\ u_1 = u_0 \text{ div } 2,\ u_2 = u_1 \text{ div } 2,\ \ldots,\ u_r = u_{r-1} \text{ div } 2 = 0$$

where u_i is the value of u after the ith iteration. Since $u_0 = n \geq 0$ and $u_i < u_{i-1}$ for $i = 1, 2, \ldots, r$, we have correctly indicated that the sequence of u_i's is finite, which is to say that the program loop terminates after r iterations for r some nonnegative integer.

```
begin {n ≥ 0}
        t := x;
        u := n;
        v := 1;
        {p: x^n = t^u v and u ≥ 0}
        while u ≠ 0 do
        begin
                if u is odd then v := t × v;
                t := t × t;
                u := u div 2;
        end
        {x^n = v}
end
```

Fig. 6 Fast program for computing x^n.

How big can r be? Since $u_r = 0 = u_{r-1}$ div 2 and $u_{r-1} > 0$, it follows that u_{r-1} must equal unity. Since $u_i \geq 2\ u_{i-1}$ [*sic*], it must be the case that $n = u_0 \geq 2^{r-1}\ u_{r-1} = 2^{r-1}$. Hence r, the number of iterations of the loop, satisfies the bound

$$r \leq (\log_2 n) + 1.$$

(One can also show that $r > (\log_2 n) - 1$—exercise for the reader.)

Thus our fast program for x^n requires a number of iterations $\approx \log_2 n$, a vast improvement over the n iterations required by the simple program of Example 2 [*sic*]. For example, for n in the billion range, the fast algorithm requires only about thirty iterations [$\log_2 10^9 = 29{,}89735\ldots$] In Fig. 7 we have presented the results of a sample execution of our fast powering algorithm.□

iteration	u	t	v
0	13	2	1
1	6	4	2
2	3	16	2
3	1	256	32
4	0	65536	$8192 = 2^{13}$

Fig. 7 Fast computation of 2^{13}.

To recap, at the heart of our development of both the simple (slow) program and the

sophisticated (fast) program for computing x^n was an invariant relation, the same one in both cases:

$$p : x^n = t^u v \text{ and } u \geq 0$$

($t = x$ in the simple case). The difference between the two programs lies solely in the way and the speed with which the variable u is forced to zero, the desired terminal value. (pp. 224-226)

Donnons ici une solution possible de l'exercice laissé au lecteur.

En notant ρ_i le reste de la division de u_{i-1} par 2 (en sorte qu'on a $u_{i-1} = 2\,u_i + \rho_i$), il vient :

$$\begin{aligned}
u_0 &= 2\,u_1 + \rho_1 \\
2\,u_1 &= 2^2\,u_2 + 2\,\rho_2 \\
2^2\,u_2 &= 2^3\,u_3 + 2^2\,\rho_3 \\
&\dots \\
2^{r-2}\,u_{r-2} &= 2^{r-1}\,u_{r-1} + 2^{r-2}\,\rho_{r-1} \\
&\text{———————————} \\
u_0 &= 2^{r-1}\,u_{r-1} + (\rho_1 + 2\rho_2 + 2^2\rho_3 + \dots + 2^{r-2}\,\rho_{r-1})
\end{aligned}$$

On a $u_0 = n$, $u_{r-1} = 1$ et $\rho_i \leq 1$ pour $i = 1, 2, \dots, r-1$. Il vient :

$$\rho_1 + 2\rho_2 + 2^2\rho_3 + \dots + 2^{r-2}\,\rho_{r-1} \leq 1 + 2 + 2^2 + \dots + 2^{r-1} = 2^r - 1$$

et donc $n \leq 2^{r-1} + 2^r - 1 < 2^{r-1} + 2^r < 2^r + 2^r = 2^{r+1}$. Il en résulte que $r + 1 > \log_2 n$ et on a donc bien $r > (\log_2 n) - 1$.

④ L'auteur conclut en ces termes :

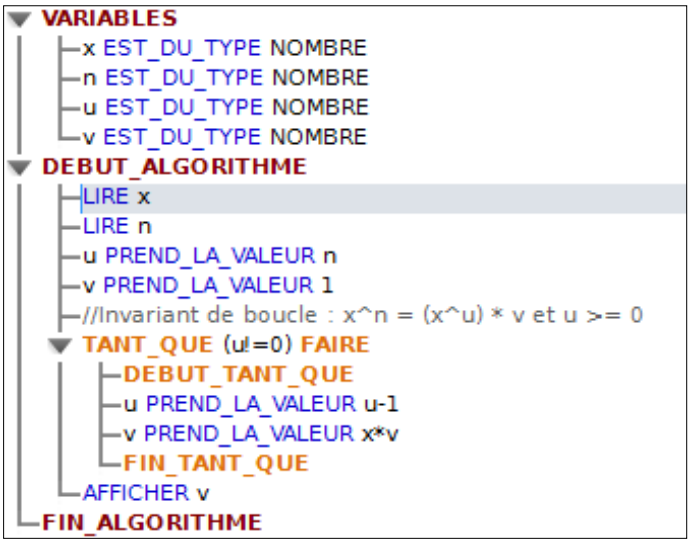
> If there is a larger message in this appendix, especially in the last two examples, it is to stress the importance of focussing on the relations between values of variables rather than the values themselves when verifying or deriving programs. As human beings we have a much better chance of grasping something static (relations among variables) than something ever changing (values of variables). ln particular, the idea of a loop invariant is the key to effective mathematical reasoning about program loops, with the framework for this reasoning being provided by the Invariant Relation Theorem. (pp. 226)

On voit ainsi que l'étude de la notion d'algorithme est bien de nature à « perturber » l'enseignement de certaines notions de mathématiques, dont celle de démonstration. Comme on l'a déjà souligné, les développements qui précèdent apportent quelques éléments à un modèle mathématique de référence *pour le travail d'analyse* de l'article étudié, notamment pour l'analyse *didactique* de cet article. Notons ici simplement que les auteurs semblent ne

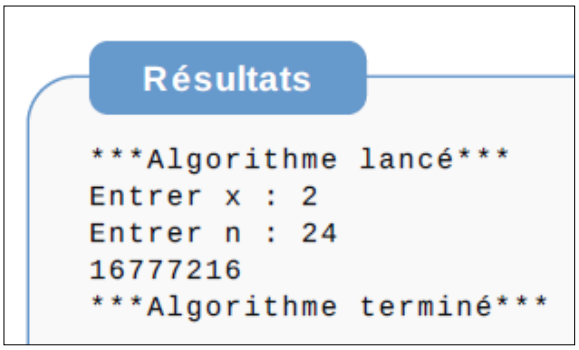
pas donner tout le relief qu'il mériterait peut-être à ce fait : contrairement à une certaine *doxa* scolaire (voire universitaire), la notion de démonstration n'est pas fixée une fois pour toutes : chaque nouveau domaine des mathématiques étudié exige un réexamen de cette notion, qui conduit à son extension à des territoires où, jusqu'alors, l'idée de démontrer n'avait pas pénétrer ou n'avait pas vraiment de formulation claire. Ainsi en va-t-il avec le domaine de l'algorithmique et la notion de preuve d'algorithme, dont nous venons de voir qu'elle peut faire l'objet de véritables théorèmes.

7. On peut illustrer les algorithmes « simple » et « rapide » proposés par Lipson à l'aide du logiciel Algobox (voir par exemple l'article « Algobox » de *Wikipédia*).

① Voici *un* programme correspondant à l'algorithme « simple » :



② Lançons le calcul pour $x = 2$ et $n = 24$; on voit s'afficher ceci :



③ Algobox peut exécuter l'algorithme pas à pas et détaille toutes les opérations réalisées, comme on le verra ci-après (toujours pour $x = 2$ et $n = 24$) ; on vérifiera que la boucle est exécutée $n = 24$ fois :

#1 Nombres/chaines (ligne 7) -> x:2 | n:0 | u:0 | v:0
#2 Nombres/chaines (ligne 8) -> x:2 | n:24 | u:0 | v:0
#3 Nombres/chaines (ligne 9) -> x:2 | n:24 | u:24 | v:0
#4 Nombres/chaines (ligne 10) -> x:2 | n:24 | u:24 | v:1
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#5 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:23 | v:1
#6 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:23 | v:2
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#7 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:22 | v:2
#8 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:22 | v:4
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#9 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:21 | v:4
#10 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:21 | v:8
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#11 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:20 | v:8
#12 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:20 | v:16
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#13 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:19 | v:16
#14 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:19 | v:32
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#15 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:18 | v:32
#16 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:18 | v:64
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#17 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:17 | v:64
#18 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:17 | v:128
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#19 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:16 | v:128
#20 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:16 | v:256
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#21 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:15 | v:256

#22 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:15 | v:512
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#23 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:14 | v:512
#24 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:14 | v:1024
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#25 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:13 | v:1024
#26 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:13 | v:2048
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#27 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:12 | v:2048
#28 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:12 | v:4096
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#29 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:11 | v:4096
#30 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:11 | v:8192
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#31 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:10 | v:8192
#32 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:10 | v:16384
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#33 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:9 | v:16384
#34 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:9 | v:32768
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#35 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:8 | v:32768
#36 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:8 | v:65536
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#37 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:7 | v:65536
#38 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:7 | v:131072
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#39 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:6 | v:131072
#40 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:6 | v:262144
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)
#41 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:5 | v:262144
#42 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:5 | v:524288
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)

#43 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:4 | v:524288

#44 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:4 | v:1048576

Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)

Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)

#45 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:3 | v:1048576

#46 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:3 | v:2097152

Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)

Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)

#47 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:2 | v:2097152

#48 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:2 | v:4194304

Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)

Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)

#49 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:1 | v:4194304

#50 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:1 | v:8388608

Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)

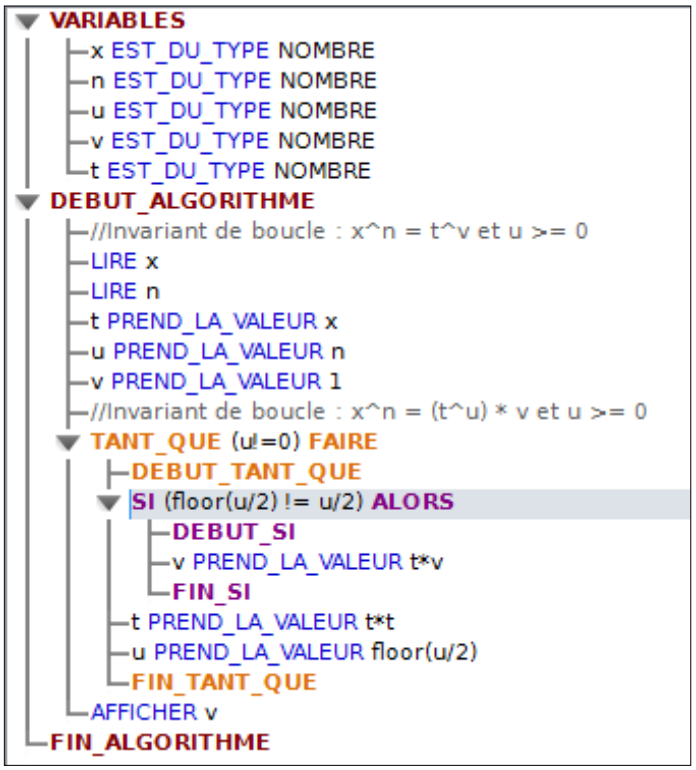
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 13)

#51 Nombres/chaines (ligne 14) -> x:2 | n:24 | u:0 | v:8388608

#52 Nombres/chaines (ligne 15) -> x:2 | n:24 | u:0 | v:16777216

Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 16)

④ Voici maintenant un programme correspondant à l'algorithme « rapide » :



⑤ L’exécution pas à pas de l’algorithme montre que, cette fois, la boucle est exécutée seulement 5 fois :

#1 Nombres/chaines (ligne 9) -> x:2 | n:0 | u:0 | v:0 | t:0
#2 Nombres/chaines (ligne 10) -> x:2 | n:24 | u:0 | v:0 | t:0
#3 Nombres/chaines (ligne 11) -> x:2 | n:24 | u:0 | v:0 | t:2
#4 Nombres/chaines (ligne 12) -> x:2 | n:24 | u:24 | v:0 | t:2
#5 Nombres/chaines (ligne 13) -> x:2 | n:24 | u:24 | v:1 | t:2
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 16)
La condition n'est pas vérifiée (ligne 17)
#6 Nombres/chaines (ligne 21) -> x:2 | n:24 | u:24 | v:1 | t:4
#7 Nombres/chaines (ligne 22) -> x:2 | n:24 | u:12 | v:1 | t:4
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 23)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 16)
La condition n'est pas vérifiée (ligne 17)
#8 Nombres/chaines (ligne 21) -> x:2 | n:24 | u:12 | v:1 | t:16
#9 Nombres/chaines (ligne 22) -> x:2 | n:24 | u:6 | v:1 | t:16
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 23)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 16)
La condition n'est pas vérifiée (ligne 17)
#10 Nombres/chaines (ligne 21) -> x:2 | n:24 | u:6 | v:1 | t:256
#11 Nombres/chaines (ligne 22) -> x:2 | n:24 | u:3 | v:1 | t:256
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 23)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 16)
La condition est vérifiée (ligne 17)
Entrée dans le bloc DEBUT_SI/FIN_SI (ligne 18)
#12 Nombres/chaines (ligne 19) -> x:2 | n:24 | u:3 | v:256 | t:256
Sortie du bloc DEBUT_SI/FIN_SI (ligne 20)
#13 Nombres/chaines (ligne 21) -> x:2 | n:24 | u:3 | v:256 | t:65536
#14 Nombres/chaines (ligne 22) -> x:2 | n:24 | u:1 | v:256 | t:65536
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 23)
Entrée dans le bloc DEBUT_TANT_QUE/FIN_TANT_QUE : condition vérifiée (ligne 16)
La condition est vérifiée (ligne 17)
Entrée dans le bloc DEBUT_SI/FIN_SI (ligne 18)
#15 Nombres/chaines (ligne 19) -> x:2 | n:24 | u:1 | v:16777216 | t:65536
Sortie du bloc DEBUT_SI/FIN_SI (ligne 20)
#16 Nombres/chaines (ligne 21) -> x:2 | n:24 | u:1 | v:16777216 | t:4.2949673e+9
#17 Nombres/chaines (ligne 22) -> x:2 | n:24 | u:0 | v:16777216 | t:4.2949673e+9
Sortie du bloc DEBUT_TANT_QUE/FIN_TANT_QUE (ligne 23)

Question 3. Quel modèle didactique de référence *pour le chercheur en didactique* relatif à l'enseignement de « l'algorithme » semble-t-il être celui des auteurs du document *E* si l'on en juge d'après les parties 2 et 3 de leur article ?

Données 3.1.

2. – L'algorithme dans l'enseignement : outil ou objet ?

2.1 *Les intérêts d'un enseignement d'algorithmique*

La commission Kahane (2000) souligne les intérêts d'« introduire une part d'informatique dans l'enseignement des sciences mathématiques et dans la formation des maîtres ». La commission avance pour cela des arguments que nous résumons ici :

• l'esprit algorithmique, présent implicitement dans l'enseignement, pourrait être travaillé et mis en lumière grâce aux instruments de l'algorithmique ;

• la programmation permet la formalisation du raisonnement ;

• les questions d'effectivité des algorithmes mettent en jeu les mathématiques ;

• traitement des données et calculs par informatique sont courants dans les autres disciplines scientifiques ;

• l'informatique a changé les mathématiques :

– en permettant d'aborder les objets sous un autre angle,

– en apportant de nouvelles questions,

– en créant de nouveaux domaines des mathématiques, aujourd'hui en plein essor,

– en transformant l'activité du mathématicien grâce à de nouveaux outils.

La commission propose des contenus pour cet enseignement d'informatique : les questions de représentation et d'approximation des nombres en informatique, les concepts de base de la programmation et de l'algorithmique.

Elle donne trois exemples : l'étude des graphes, la recherche d'enveloppes convexes dans le plan et l'étude des fractions continues.

Il semble que cette proposition ait été entendue au vu des programmes de seconde pour la rentrée 2009.

2.2 *Place actuelle et rôle de l'algorithme dans les programmes et les manuels* [7. Avant application des programmes de seconde à la rentrée 2009.]

Nous avons souligné plus haut que considérer l'algorithme sous son aspect objet conduit à se poser les questions de la construction, la preuve, la complexité et l'optimalité des algorithmes considérés. Nous ne nous intéresserons donc ici qu'aux algorithmes permettant un questionnement sur au moins un de ces points (au regard de l'algorithmique comme savoir

savant et non du point de vue des programmes ou des manuels). Pour préciser notre démarche, regardons sur des exemples d'algorithmes issus des programmes si ces questions sont en jeu.

L'algorithme d'Euclide soulève clairement toutes ces questions, nous nous intéresserons donc à lui dans les programmes et manuels. C'est aussi le cas des algorithmes de recherche par dichotomie ou du pivot de Gauss. Les questions de tri de données permettent aussi de tels questionnements.

Par contre, bien que les résolutions d'équations du premier degré ou du second degré avec le discriminant puissent être vue [*sic*] comme des algorithmes, elles ne soulèvent pas ces questions de construction, de preuve et de complexité d'algorithmes. Ce sont en quelque sorte des formules, que l'on peut appliquer de manière systématique et implémenter informatiquement mais qui n'interrogent pas l'objet mathématique algorithme. Nous laisserons ces « algorithmes » de côté dans nos analyses des programmes et des manuels.

Dorénavant, si cela n'est pas précisé, lorsque nous emploierons le terme algorithme, nous exclurons les formules. Nous entendrons par algorithmique la discipline du savoir savant qui étudie l'algorithme en tant qu'objet mathématique. Lorsque nous parlerons par exemple, d'introduire des notions d'algorithmique, nous entendons par là un questionnement sur la construction d'algorithmes, leur preuve, l'étude de leur complexité etc.

Ces précisions faites, nous pouvons étudier la place et le rôle de l'algorithme (formules exclues, donc) dans les programmes et les manuels. Cette étude s'est appuyée sur la dialectique outil-objet (Douady, 1986), sur une approche praxéologique des contenus et sur une analyse écologique des savoirs (Chevallard, 1998). Nous relatons ici les principaux résultats de cette étude (pour plus de détails, voir Modeste (2009)).

Programmes de lycée

La place de l'algorithme n'est spécifiée que dans les programmes de première et terminale L dans le cadre de l'option de mathématiques, et en terminale ES dans le cadre de la spécialité mathématiques. Dans le cas de la série littéraire, l'algorithmique est un domaine transversal qui doit permettre d'aborder les mathématiques de manière pratique et concrète et de distinguer résolutions théorique et effective de problèmes. Les élèves doivent être entraînés à des techniques relatives à l'algorithmique : décrire, interpréter et mettre en œuvre des algorithmes simples. Cela semble s'articuler avec l'enseignement obligatoire de mathématiques-informatique (notamment concernant l'utilisation des outils informatiques). On est proche d'un enseignement de mathématiques appliquées.

Dans la spécialité mathématiques de terminale ES, l'algorithmique est amenée par la théorie des graphes. L'algorithme est présenté comme un outil indispensable pour traiter des graphes complexes, certains algorithmes doivent être présentés mais la majorité des exercices doit se

résoudre sans recours à l'algorithme. Ici aussi, l'accent est mis sur l'aspect appliqué des mathématiques, mais cela passe plutôt par la modélisation.

Dans les deux cas, aucun exposé théorique d'algorithmique ne doit être fait.

Dans la série scientifique, l'algorithme est très peu présent et ne fait l'objet d'aucun travail particulier. Au vu de ce qui est dit de l'algorithmique et son intérêt, dans les programmes de première et terminale L, on peut s'interroger sur son absence dans la série S. On peut faire l'hypothèse que le constat serait le même concernant la place des mathématiques appliquées.

Dans les trois filières, seul l'aspect outil de l'algorithme est mis en valeur par les programmes (avec une légère différence pour la série L, où la question de la complexité est évoquée à propos de l'algorithmique mais ne réapparaît pas dans les contenus).

D'autre part, on constate que trois domaines sont privilégiés par les programmes pour l'introduction d'algorithmes :

– L'arithmétique (collège, seconde, première et terminale L (option), terminale ES (spécialité)).

– L'analyse, dans le cas d'approximations (première L (option) et terminale S).

– La théorie des graphes (Terminale ES (spécialité)).

Malgré cela, dans ces domaines, beaucoup d'algorithmes sont évités alors que tout est favorable à leur introduction. C'est notamment le cas en arithmétique à différents niveaux ou en spécialité de terminale ES, dans le cadre de la théorie des graphes, où l'on peut s'interroger sur ce qui a motivé le choix des algorithmes au programmes [8. Voir (Cartier, 2008).] On note aussi que d'autres domaines pourraient être des champs d'introduction d'algorithmes : simulation et modélisation, géométrie, résolution de systèmes d'équations, calcul matriciel.

Manuels du lycée

Nous avons choisi d'analyser un manuel pour chaque classe des filières générales du lycée. Pour la seconde et la filière S, nous avons choisi la collection Terracher (Hachette).

Mais cette collection n'existe pas pour les autres filières. Pour la filière ES, nous avons choisi les manuels de la collection Transmath (Nathan). Pour la spécialité de la filière L, il semble qu'il n'y ait pas de manuels. Pour répondre à nos questions, concernant cette filière, une étude des pratiques enseignantes en classe aurait été nécessaire. Nous n'avons pas pu la mettre en œuvre ici.

Dans les manuels étudiés, nous avons fait les constats suivants :

– Le concept d'algorithme est présent mais très peu mis en valeur institutionnellement. D'une part, l'algorithme n'est pas complètement défini, voire souvent pas du tout défini [9. La seule définition de l'algorithme que nous avons rencontré [*sic*] dans ces manuels le présente comme une « méthode permettant de résoudre un problème et qui se présente sous la forme d'une suite d'opérations élémentaires obéissant à un enchaînement déterminé ».] D'autre part, il ne fait pas

l'objet d'une présentation spécifique et les algorithmes sont souvent présents de manière implicite sans qu'un travail spécifique ne soit réalisé (travail de description, de preuve ou d'analyse).

– Les algorithmes proposés dans la partie « cours » sont rarement prouvés rigoureusement ; souvent, on n'a pas même l'ébauche d'une preuve. Cependant, deux domaines mathématiques laissent une place à la preuve : l'arithmétique en spécialité de terminale S et la théorie des graphes en spécialité de terminale ES où quelques algorithmes sont prouvés ou justifiés (notamment l'algorithme d'Euclide pour le *pgcd* et le théorème d'existence d'un cycle eulerien dans un graphe), mais où d'autres preuves pourraient être traitées.

– La complexité des algorithmes de la partie « cours » n'est questionnée dans aucun manuel et de manière extrêmement rare dans les exercices (il s'agit alors de compter des multiplications ou des additions).

– Des algorithmes sont très rarement mobilisés dans des preuves. Cela est tout de même présent de manière sporadique dans les spécialités de terminales S et ES 10 (preuve de l'existence des coefficients de Bezout ou preuve du théorème d'existence de chemins eulériens).

– Un unique type de tâche est réellement développé relativement à l'algorithme : la mise en œuvre sur des données, c'est-à-dire un type de tâche relatif à l'*aspect outil*.

Nous pouvons dire que l'algorithme occupe strictement la même place que celle prévue dans les programmes. Seul son aspect outil est développé, comme si l'algorithme n'avait qu'une existence en tant que technique de résolution. Quant au lien de l'algorithme à la preuve, il est très peu exploité par les manuels.

2.3 *En conclusion*

Nous avons vu que l'algorithme est présent dans le savoir à enseigner, mais sa place et son rôle sont très restreints et son lien à la preuve n'est pas exploité : sa place est limitée à certains domaines mathématiques (arithmétique, approximation en analyse et théorie des graphes) où il n'apparaît essentiellement que comme outil. Nous avons montré dans la première partie, que l'algorithme en tant qu'objet aurait toute sa place dans l'enseignement, en particulier pour son lien à la preuve. Actuellement, son rôle est limité à celui d'outil, comme nous l'avons montré dans cet état des lieux alors qu'il existe d'autres aspects de l'algorithme qui pourraient être étudiés (la complexité par exemple) et qui ne sont pas du tout traités dans les programmes et manuels.

Le lien de l'algorithme à la preuve n'est ni évoqué dans les programmes ni exploité par les manuels : on ne rencontre pas beaucoup de preuves d'algorithme et les algorithmes sont rarement mis en lien avec les preuves. En résumé, dans les programmes et les manuels de l'enseignement secondaire français, il semble que l'algorithme ne vit que dans la niche

technique de résolution. Mais d'autres niches existent pour lui dans le savoir savant : la niche preuve et bien sûr la niche algorithmique elle-même.

Éléments d'analyse 3.1

1. Dans ce qui suit, nous essayons de dégager quelques éléments entrant dans le *modèle didactique de référence* (MDR) que l'on peut raisonnablement prêter aux chercheurs-auteurs, tels qu'ils apparaissent dans les « données » proposées. Il s'agit donc d'un MDR *possible*, qui nous paraît *compatible* avec l'article examiné. Il conviendra toujours d'entendre les assertions, principes, préceptes, etc., dégagées dans ce cadre comme des éléments conjecturaux non inconciliables avec ce qui pourrait être le MDR des auteurs, sous réserve, et jusqu'à preuve du contraire.

2. Les auteurs examinent les modes de présence de ce qu'ils nomment « l'algorithme » dans l'enseignement des mathématiques au secondaire français. En d'autres termes, il semble qu'ils s'efforcent d'identifier certaines *conditions et contraintes* gouvernant, à l'époque de leur recherche, *l'écologie et l'économie* « scolaires » d'éléments praxéologiques ayant trait aux algorithmes et à l'algorithmique, dans l'enseignement des mathématiques au secondaire français.

① Cet examen porte d'abord sur une production de la noosphère de ce système, le rapport de la « Commission de réflexion sur l'enseignement des mathématiques » dirigée par Jean-Pierre Kahane. Ce rapport a été publié en 2002 par l'éditrice Odile Jacob (pour plus de renseignements, on se reportera à http://www.cfem.asso.fr/ressources/rapports-enseignement-mathematiques/commission-kahane). Les auteurs se réfèrent plus précisément au texte intitulé « Informatique et enseignement des mathématiques », qu'on trouvera en ligne à l'adresse http://smf4.emath.fr/en/Enseignement/CommissionKahane/RapportInfoMath/RapportInfoMath.pdf.

② Comme il en va pour l'ensemble de l'enquête dont rend compte cette partie 2 de l'article, la signification d'un tel examen en termes de MDR n'est pas évidente. L'objet en est peut-être de s'assurer que les contenus d'enseignement censément nouveaux mis au cœur de leur recherche par les auteurs ne sont pas en fait *déjà présents*, au moins partiellement, dans l'enseignement secondaire des mathématiques.

❶ Il y a là un aspect que l'on retrouve à l'œuvre assez fréquemment dans les recherches en didactique : des contenus d'enseignement ne font l'objet d'une recherche que si ces contenus ne sont pas *déjà*, et depuis longtemps peut-être, enseignés. C'est souvent le nouveau, le pas-encore-enseigné qui aimante les chercheurs, du moins s'ils adhèrent à ce précepte implicite d'un certain MDR professoral (que nous mettrons dans le MDR conjectural des auteurs) : *ce qui est anciennement enseigné n'appelle pas en principe de réorganisation didactique* ⟨1⟩. Ce principe est remis en cause, au mieux, lorsque l'enseignement « ancien » semble être devenu fortement inadéquat (parce que les publics d'élèves ont changé, parce que certains environnements ont évolué, etc.). Les contenus d'enseignement « anciens » semblent ainsi ne pouvoir faire l'objet, au mieux, que de recherches « de second choix ».

❷ Là encore, on peut soupçonner l'effet d'une *rupture insuffisante* entre la position de didacticien et la position de professeur : ne s'intéresser – avec angoisse ou au contraire avec excitation – qu'au nouveau est plutôt une attitude de professeur. (De la même façon, les médecins s'inquiètent des pathologies nouvelles ; les anciennes, ils les soignent depuis longtemps…) S'il n'y avait pas véritablement *nouveauté*, les auteurs devraient donc, pour faire apparaître leur travail comme relevant de la *recherche*, faire leur et défendre une autre problématique en matière de recherche, qui supposerait un autre MDR, intégrant par exemple le principe du caractère problématique de l'enseignement courant de tels ou tels contenus.

③ L'objectif de l'examen auquel les auteurs procèdent pourrait être aussi – et aussi bien – d'analyser les conditions et contraintes prévalentes pour voir s'il existerait une possibilité de modifier l'état actuel du système dans le sens qu'ils envisagent, celui de l'avènement de « l'algorithme » en tant qu'*objet* mathématique, et non comme « simple » outil du travail mathématique (à la manière d'un rapporteur, d'un compas ou d'une calculatrice), une entité mathématique posant, en l'espèce, des problèmes de construction, de preuve et de complexité.

④ L'enquête – la lecture de la partie du rapport Kahane indiquée plus haut – leur révèle que la Commission souhaite voir « introduire une part d'informatique dans l'enseignement des sciences mathématiques et dans la formation des maîtres ». (On aura noté que cette part d'informatique doit prendre place, apparemment, *dans le cadre même de l'enseignement des mathématiques*.)

❶ Les raisons de ce souhait de la Commission sont multiples. La principale sans doute, aux yeux des auteurs, est celle-ci : « l'esprit algorithmique, présent implicitement dans

l'enseignement, pourrait être travaillé et mis en lumière grâce aux instruments de l'algorithmique. » On notera l'ambiguïté toute classique (au moins en mathématiques) du propos : d'une part, il s'agirait de « travailler », non pas l'algorithmique, mais *l'esprit algorithmique* ; d'autre part, ce travail se réaliserait tout de même à l'aide des *outils* de l'algorithmique, parmi lesquels, peut-on penser, « la programmation », qui doit permettre « la formalisation du raisonnement ».

❷ Cela noté, on peut imaginer que le diagnostic de la Commission, tel du moins que le restituent les auteurs, est de nature à conforter la problématique de ces auteurs. L'enseignement des mathématiques serait actuellement pauvre en contenus algorithmiques – qui y sont, au mieux, « implicites » – et l'enrichissement de tels contenus ne saurait être que *bénéfique* aux mathématiques enseignées. Les auteurs ne se départissent pas d'un point de vue qui, semble-t-il, participe de leur MDR : *l'introduction d'éléments nouveaux dans un corpus (mathématique) enseigné se justifie d'abord par le souci de revitaliser, de redynamiser ce corpus et son enseignement* ⟨2⟩, plutôt que par l'intérêt de diffuser des connaissances *nouvelles* au nom de leur *utilité* supposée. L'introduction d'éléments d'algorithmique permettrait « d'aborder les objets sous un autre angle » et cela « en apportant de nouvelles questions ». C'est un peu comme si l'on avait autrefois introduit des rudiments de calcul différentiel au lycée, non pas pour leur intérêt propre, mais pour tenter de revivifier l'algèbre classiquement enseignée, supposée alors à bout de souffle !

⑤ Nous avancerons encore que, dans le MDR des auteurs, *une novation est d'autant plus justifiée qu'elle est souhaitée par davantage d'autorités* ⟨3⟩, à quelque niveau que se situent ces autorités.

❶ En l'espèce, cette attitude se voit confirmée par une décision récente au moment où les auteurs rédigent leur article : l'algorithmique est officiellement introduite dans le nouveau programme de mathématiques de seconde qui entre en vigueur à la rentrée 2009 (ce programme a été publié dans le *Bulletin officiel de l'Éducation nationale* n^{o} 30 du 23 juillet 2009). Les auteurs disent voir là une prise en compte des recommandations du rapport Kahane, ce qui donne un peu plus de crédit à l'hypothèse ⟨3⟩, formulée ci-dessus, selon laquelle, dans le MDR des auteurs, et classiquement, « une novation est d'autant plus justifiée qu'elle est souhaitée par davantage d'autorités ».

3. Les auteurs passent ensuite à l'examen des contenus qui, dans l'état de choses prévalent *avant* l'entrée en vigueur du nouveau programme de 2de, peuvent être reliés à l'étude de « l'algorithme » tel qu'ils l'entendent, c'est-à-dire « sous son aspect objet ».

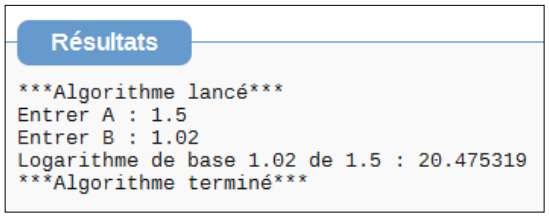
① On doit noter ici un nouvel élément du MDR que l'on peut prêter aux auteurs : *l'entrée d'une entité (supposée) mathématique dans le corpus enseigné est d'autant mieux justifiée qu'apparaît plus substantielle, mathématiquement, l'organisation praxéologique où elle est censée venir prendre place* ⟨4⟩. Dans le cas de « l'objet algorithme », cette organisation doit, selon les auteurs, répondre aux « questions de la construction, la preuve, la complexité et l'optimalité des algorithmes considérés ».

② Les auteurs mentionnent des algorithmes présents dans le cursus mathématique secondaire qui pourraient nourrir l'étude de « l'algorithme » : « Pour préciser notre démarche, écrivent-ils, regardons sur des exemples d'algorithmes issus des programmes si ces questions sont en jeu. » Ils repèrent ainsi plusieurs algorithmes ou types d'algorithmes : algorithme d'Euclide, recherches par dichotomie, pivot de Gauss, algorithmes de tri.

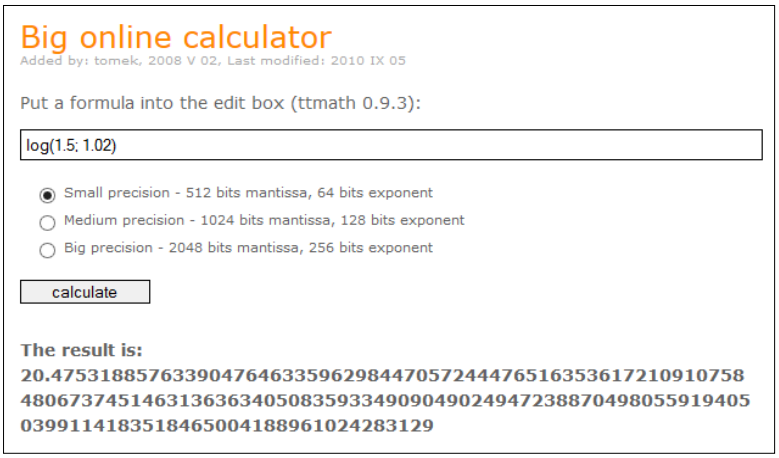
③ On doit souligner encore la décision des auteurs d'ignorer ces « algorithmes » qui dériveraient simplement de « l'application d'une formule », comme il en va avec « les résolutions d'équations du premier degré ou du second degré avec le discriminant ». Ce refus est clair et net et fait l'objet de déclarations réitérées : « Nous laisserons ces "algorithmes" de côté dans nos analyses des programmes et des manuels. » (On aura noté les guillemets mis au mot *algorithme*.) La raison de ce rejet paraît claire : ces algorithmes, selon les auteurs, ne sont que des *outils* à utiliser, non des *objets* à étudier mathématiquement.

❶ Prenons un exemple. Supposons que l'on considère une dette D qui croît de 2 % par période. À l'instant 0, elle vaut D_0 ; au bout d'une période, elle vaut $D_1 = 1{,}02 \times D_0$ et, au bout de n périodes, elle vaudra $D_n = (1{,}02)^n \times D_0$. On veut savoir au bout de combien de périodes elle aura augmenté de 50 % par exemple. L'équation à résoudre en n est : $(1{,}02)^n \times D_0 = 1{,}5 \times D_0$, soit donc $(1{,}02)^n = 1{,}5$. Prenons le logarithme de base 1,02 de chacun des deux membres ; il vient : $n \log_{1{,}02} 1{,}02 = \log_{1{,}02} 1{,}5$, c'est-à-dire $n = \log_{1{,}02} 1{,}5$.

❷ Il est bon alors de pouvoir calculer le logarithme de base *b quelconque* d'un nombre *a*. On peut pour cela programmer dans Algobox un tel algorithme, fondé sur la *formule* $\log_b a = \frac{\log a}{\log b}$; on obtient ainsi :



Bien entendu, un tel algorithme pose un problème de construction, un problème de correction, un problème de complexité et d'optimalité, même s'il ne s'agit là, en l'espèce, que de problèmes jugés « faciles ». En même temps, il est vrai que, si l'on disposait d'une calculatrice ayant une touche $\log_b a$, cela ferait aussi bien l'affaire et ce serait là, alors, un pur outil, qui plus est mathématiquement opaque (ce qui dispense notamment de connaître la formule rappelée ci-dessus). Ainsi en va-t-il par exemple avec le *Big Online Calculator*, calculatrice en ligne que l'on trouvera à l'adresse http://www.ttmath.org/online_calculator et dont voici une capture d'écran :



④ Que peut-on induire de tout cela s'agissant du MDR que l'on pourrait prêter aux auteurs ? Nous y verrons la manifestation d'un principe contestable mais très répandu chez les chercheurs en didactique des mathématiques et que l'on peut énoncer ainsi : *une organisation didactique a pour objet essentiel d'organiser la première rencontre avec les notions à enseigner* ⟨5⟩, soit ici les notions solidaires d'algorithme, de construction d'un algorithme, de preuve d'algorithme, de complexité et d'optimalité d'un algorithme.

❶ En particulier, on ne se préoccupe que fort rarement, dans les recherches communes en didactique, de la prise en charge théorique et pratique du moment du *travail de la technique* ou encore du moment *de l'évaluation*. Le moment de la première rencontre est en quelque sorte regardé comme le « morceau noble », la plupart des autres moments comme des « bas morceaux » : c'est là une frontière actuelle de la science didactique, ce qui, par exemple, laisse le champ libre aux « spécialistes » non didacticiens de l'évaluation.

❷ Cette *réduction didactique* accompagne une opération qu'on peut décrire comme ayant pour but l'amorce de la transposition didactique d'un savoir « savant » que les auteurs désignent sans ambiguïté : « Nous entendrons par algorithmique la discipline du savoir savant qui étudie l'algorithme en tant qu'objet mathématique. » En accord avec ce qui précède, ils ajoutent : « Lorsque nous parlerons par exemple d'introduire des notions d'algorithmique, nous entendons par là un questionnement sur la construction d'algorithmes, leur preuve, l'étude de leur complexité, etc. » Cela confirme à la fois que l'enseignement d'un certain complexe d'organisations mathématiques est visé *et que* le problème de cet enseignement est formulé en ne tenant compte que des composantes praxéologiques « nobles ».

4. La suite de la partie 2 de l'article est un compte rendu de l'enquête qu'ont réalisée les auteurs sur « la place et le rôle de l'algorithme (formules exclues, donc) dans les programmes et les manuels ». Le bilan de cette enquête est, en apparence, décevant : « Dans les trois filières, seul l'aspect outil de l'algorithme est mis en valeur par les programmes (avec une légère différence pour la série L…) » Mais le sens donné à un tel constat reste à préciser.

① Les auteurs observent que « beaucoup d'algorithmes sont évités alors que tout est favorable à leur introduction ». En d'autres termes, il y aurait un « potentiel » de contenus déjà là mais laissés en friche. Les auteurs notent encore : « C'est notamment le cas en arithmétique à différents niveaux ou en spécialité de terminale ES, dans le cadre de la théorie des graphes, où l'on peut s'interroger sur ce qui a motivé le choix des algorithmes au programmes… » À cette exploitation limitée de ressources « naturelles » s'ajoute la non-exploitation de certains domaines riches de possibilités : « On note aussi, poursuivent les auteurs, que d'autres domaines pourraient être des champs d'introduction d'algorithmes : simulation et modélisation, géométrie, résolution de systèmes d'équations, calcul matriciel. »

② Derrière l'insistance mise sur la double condition d'un *riche potentiel mal exploité* (voire quasiment inexploité), on peut conjecturer que le MDR dont nous traçons un portrait robot

contienne le postulat suivant : *l'introduction d'organisations mathématiques dans un cursus d'études est facilitée par le double fait que les contenus à introduire figurent déjà, en germe, dans le cursus actuel et, en même temps, s'y trouvent traditionnellement négligés* ⟨6⟩, en sorte que l'innovation envisagée promet une amélioration substantielle, de tel ou tel point de vue. Les auteurs notent que, dans les filières L et ES, « aucun exposé théorique d'algorithmique ne doit être fait », ce qui est classique mais peut se révéler invalidant. Pire encore, en série S, « l'algorithme est très peu présent et ne fait l'objet d'aucun travail particulier ».

③ L'examen de divers *manuels* (pp. 59-60) confirme abondamment ce que l'analyse des programmes permet de conclure. On aura noté que les auteurs ne vont pas jusqu'à l'observation des « pratiques enseignantes en classe », comme s'ils admettaient ce principe (que les professeurs ne partagent sans doute pas) : *une organisation didactique et les contenus de connaissance dont elle organise l'étude sont essentiellement déterminés (« vers le haut », sinon « vers le bas ») par le contenu des programmes et par le contenu des manuels relatifs aux thèmes d'étude considérés* ⟨7⟩.

5. La conclusion de la partie 2 semble confirmer les éléments du MDR que nous avons induits des développements précédents.

① Les auteurs rappellent d'abord que, si « l'algorithme est présent dans le savoir à enseigner », en revanche « sa place et son rôle sont très restreints et son lien à la preuve n'est pas exploité ». Par contraste, ils soulignent l'apport possible de leur problématique de recherche, du fait que, selon eux, « l'algorithme en tant qu'objet aurait toute sa place dans l'enseignement, en particulier pour son lien à la preuve ». On retrouve en filigrane, ici, ce principe : *un projet de rafraîchissement d'un cursus d'études est d'autant mieux fondé qu'il ambitionne non pas tant d'introduire du nouveau et de l'utile que de fortifier des notions emblématiques – et donc « anciennes » – de ce cursus d'études* ⟨8⟩.

② Dans le cas étudié, cette « valeur » à laquelle s'articule le projet des auteurs est « la preuve ». On a déjà noté le déplacement lexical adopté : alors que l'on parle traditionnellement, en français, de *démonstration*, ce mot est ici remplacé par *preuve*, traduction de l'anglais *proof* qui s'est répandu dans le français de l'informatique, sans doute par le biais du syntagme *proof of a program*, traduit littéralement par « preuve de programme ». Le mot *démonstration* apparaît trois fois dans l'article, et le verbe *démontrer* une fois. En revanche, le mot *preuve* apparaît 86 fois (en excluant ses occurrences dans le titre

courant et dans la bibliographie) et le verbe *prouver* 14 fois. Soulignons que *preuve* apparaît même dans des contextes où l'emploi de *démonstration* est habituel, par exemple dans cette note de bas de page : « Pour une preuve, voir Modeste (2009). »

Données 3.2

3. – Une situation pour l'algorithme

Nous souhaitons montrer ici que l'on peut proposer des situations permettant l'accès aux différents aspects de l'algorithme et notamment à l'aspect objet. Nous allons, pour cela, nous intéresser à un problème de recherche de fausses pièces.

3.1 *Présentation du problème*

Un problème de fausses pièces

Dans un ensemble de pièces, indiscernables à la vue ou au toucher, se trouvent des fausses pièces. Les vraies pièces pèsent toutes le même poids, les fausses aussi mais leur poids est différent de celui des vraies. A l'aide d'une balance Roberval à deux plateaux et sans poids, comment peut-on retrouver les fausses pièces ? Quelle est la méthode qui permet de les retrouver en effectuant le moins possible de pesées ?

On peut définir plusieurs variantes de ce problème en jouant sur les variables de la situation suivantes :

- Poids des fausses pièces :

– On sait au départ que les fausses pièces sont plus lourdes que les vraies (ou plus légères).
– On ne sait rien.

- Le nombre de fausses pièces :

– Fixé à l'avance.
– Compris dans un intervalle donné.
– Non connu à l'avance.

Ce problème ainsi que la plupart de ses variantes, ne sont pas encore complètement résolus pour la recherche en mathématiques [Note 11. On pourra consulter, entre autres, (Tosic, 1983), (Pyber, 1986) ou (Aigner, Li, 1997).] Nous limiterons notre étude à quelques variantes, en jouant sur les variables didactiques, que nous choisirons pour permettre une mise en œuvre d'algorithmes et leur étude par des élèves.

L'algorithmique dans ce problème

Ce problème relève bien d'une problématique d'algorithmique : il s'agit de construire un algorithme de recherche des fausses pièces, de le prouver et d'étudier son optimalité. L'énoncé du problème ne fait appel à aucune notion mathématique particulière mise [*sic*] à part la notion d'optimalité, mais qui peut-être comprise au sens intuitif. L'appropriation de ce problème nous paraît donc relativement facile.

De plus, la construction d'une méthode, même grossière du point de vue de la complexité, peut être mise en place. Pour prouver qu'une méthode de recherche donnée trouve toujours les fausses pièces, un raisonnement s'appuyant sur une logique intuitive suffit la plupart du temps (disjonction de cas par exemple).

La question la plus délicate semble donc être celle de l'optimalité. En effet, elle soulève le problème du choix d'un critère d'optimalité. Deux critères peuvent être utilisés : la complexité au pire et la complexité en moyenne.

La complexité au pire est le nombre de pesées qu'effectue un algorithme dans le pire des cas. Autrement dit, la complexité au pire pour un algorithme donné est le nombre maximum d'étapes (ici les pesées) qu'il effectue, maximum calculé sur l'ensemble des instances d'une taille donnée (ici pour un nombre fixé de pièces). La complexité en moyenne est la moyenne des nombres d'étapes effectuées par l'algorithme, moyenne calculée sur l'ensemble des instances d'une taille donnée (généralement on suppose les instances équiprobables). Elles s'expriment toutes les deux en fonction de la taille de l'instance, c'est-à-dire ici, en fonction du nombre de pièces. Ici, nous ne nous intéresserons pas au problème de l'optimalité pour la complexité en moyenne. Cela ne veut pas dire que cette question n'est pas intéressante du point de vue mathématique ou didactique.

Malgré ce choix, nous le verrons dans l'analyse mathématique du problème, ce sont les questions de complexité et d'optimalité qui peuvent demander le plus d'outils mathématiques (récurrence, puissances, logarithmes,...).

3.2 *Etude mathématique du problème*

Le problème des pesées est un problème d'optimisation. La résolution de tels problèmes repose sur l'étude de deux sous-problèmes :

- P_M : Construire une méthode de recherche de la fausse pièce pour un problème donné
- P_O : Prouver l'optimalité de la méthode. On distinguera parmi les stratégies de résolution de P_O celles qui relèvent d'un argument d'optimalité locale et celles qui relèvent d'un argument d'optimalité globale.

P_M permet de travailler sur la condition suffisante et P_O sur la condition nécessaire. Parfois les deux sous-problèmes sont traités simultanément : on construit une méthode et sa construction prouve son optimalité. Nécessairement cela s'appuiera sur un argument d'optimalité locale. Pour le problème d'optimalité P_O on peut définir le problème dual P_O* : Si l'on fixe un nombre de pesées *p*, combien de pièces peut-on traiter avec *p* pesées ? Parfois le passage à P_O* pourra s'avérer fructueux pour répondre à P_O.

On s'intéresse maintenant aux différentes stratégies envisageables en fonction des variables suivantes : le nombre de fausses pièces et les informations sur leur poids :

– Le nombre de fausses pièces peut être connu (*1, 3,* compris entre *0* et *2*...) ou non.

– Le poids de la fausse pièce peut être connu (plus lourd par exemple) ou non.

Dans notre étude on se limitera aux cas où le nombre de fausses pièces n'excède pas un : c'est-à-dire lorsqu'il y a exactement une fausse pièce ou au plus une fausse pièce. On s'intéresse donc aux quatre problèmes suivants :

- $P_{1,+}$: 1 fausse pièce, plus lourde.
- $P_{1,+/-}$: 1 fausse pièce, plus lourde ou plus légère.
- $P_{0/1,+}$: 0 ou 1 fausse pièce, plus lourde.
- $P_{0/1,+/\_}$: 0 ou 1 fausse pièce, plus lourde ou plus légère.

On présentera ici les stratégies et les résultats mathématiques pour $P_{1,+}$. Pour les autres cas on pourra consulter (Modeste, 2009).

Stratégies pour $P_{1,+}$:

Dans une démarche de recherche on est souvent amené à étudier des petits cas afin d'essayer de mieux comprendre l'enjeu du problème et de pouvoir, par exemple, émettre des conjectures. Cela donne lieu à la stratégie suivante :

- Stratégie expérimentale S_{EXP} :

On traite les petits nombres de pièces (par exemple par énumération de cas). Cela peut aboutir à des « généralisations » formulées comme conjectures. D'autre part, pour un nombre de pièces fixé, si l'on a traité tous les cas plus petits, on peut énumérer toutes les pesées possibles (de manière plus ou moins organisée) et déduire la méthode optimale pour ce cas. Cette méthode s'appuie sur un argument d'optimalité locale. Elle devient très vite fastidieuse mais permet d'avoir un grand nombre d'exemples sur lesquels s'appuyer pour faire des conjectures ou les invalider.

Pour répondre à ces conjectures, il convient de formaliser ce qu'est une pesée. Un premier point de vue est qu'effectuer une pesée, c'est prendre deux ensembles de pièces de même taille et de les comparer pour savoir lequel des deux est le plus lourd. Cette vision peut donner lieu à la stratégie S_{DICHO}.

• Stratégie de dichotomie S_{DICHO} :

Pour traiter *n* pièces, on compare « la moitié » des pièces sur chaque plateau. Le déséquilibre permet de garder *[n/2]* pièces à tester [Note 12. On notera entre crochets *[ ]* la partie entière inférieure.], on réitère ensuite le processus. Si *n* est impair, un équilibre de la balance est possible et indique que la fausse pièce est celle laissée de côté. Cela donne un algorithme en $[log_2(n)]$ pesées au pire.

Cette stratégie ne conduit pas à un algorithme optimal, il suffit pour s'en convaincre de regarder le problème pour *9* pièces. Avec S_{DICHO} il faudra *3* pesées alors que S_{EXP} donne une méthode en *2* pesées. L'optimalité de cet algorithme peut reposer sur la conception erronée : plus on pèse de pièces à la fois, plus on est efficace (argument d'optimalité locale). En fait, cette méthode ne tient pas assez compte du fait qu'une pesée donne aussi de l'information sur l'ensemble des pièces non pesées.

En effet, effectuer une pesée, c'est faire trois tas : deux de même taille *A* et *B* que l'on compare, et un tas *C* que l'on laisse de côté. Trois résultats sont alors possibles :
– A est plus léger que B et alors A contient la fausse pièce [*sic*].
– A est plus lourd que B et alors B contient la fausse pièce [*sic*].
– A et B sont de même poids et c'est le tas C qui contient la fausse pièce.

En s'appuyant sur cette observation, on peut mettre en oeuvre la stratégie suivante.

• Stratégie de trichotomie S_{TRICHO} :

On divise le tas de pièces restantes en trois paquets de « même taille » i.e. tels que
$|A| = |B|$ et $| |A|-|C| | \leq 1$.
Pour *n* pièces, avec $3^{k-1} < n \leq 3^k$, cet algorithme demande *k* pesées, autrement dit $[log_3(n)]$ pesées ou encore cette méthode permet de traiter en *k* pesées jusqu'à 3^k pièces.

Proposition. Cet algorithme est optimal pour la complexité au pire [Note 13. Pour une preuve, voir Modeste (2009).]

Une autre façon de concevoir une pesée, est de n'interpréter le résultat que comme un équilibre ou un déséquilibre, sans tenir compte de quelles pièces sont sur le plateau le plus lourd. Cela peut conduire à la stratégie :

• Stratégie par étalon S_{ETAL} :

On utilise un ensemble *E* de pièces identifiées comme vraies pour tester un ensemble *A* de pièces inconnues par comparaison. Soit les pièces de *A* sont vraies et elles s'ajoutent à l'ensemble *E* des pièces talons, soit on sait que *A* contient la fausse pièce et il ne reste plus qu'à la retrouver. Pour ce faire on peut, par exemple, identifier des bonnes pièces dans *A* grâce à *E* et les enlever jusqu'à n'en avoir plus qu'une.

• Stratégie de passage par un autre problème S_{PROB} :

On se ramène à des problèmes déjà traités ou qui paraissent plus simples (voire on déduit de leur optimalité celle de la méthode construite).

3.3 *Analyse de la situation*

Une Situation de Recherche en Classe

Pour observer la construction d'algorithmes, leur preuve ainsi que l'étude de leur complexité et de leur optimalité, il est intéressant de mettre les apprenants en situation de résolution de problème en groupe, face à un problème pour lequel ils peuvent adopter une démarche proche de celle du chercheur (essais-erreurs, conjectures, preuve, simplification du problème, ...). Les Situations de Recherche en Classe, dites SiRC, développées et étudiées par l'équipe Maths à Modeler semblent se prêter particulièrement bien à ce type d'études et le problème des pesées nous paraît complètement adapté à ce cadre. D'après les travaux de (Grenier, Payan, 2003), (Godot, Grenier, 2004) et (Cartier et al., 2006), les SiRC se caractérisent par les aspects suivants :

a) Le problème abordé doit être proche de la recherche actuelle.

b) La question initiale doit être facile d'accès et ne doit pas être formalisée en termes mathématiques pour faciliter l'entrée dans le problème.

c) Des stratégies d'approche simples doivent exister pour permettre d'entrer dans une démarche de recherche.

d) De nombreuses stratégies de recherche peuvent être mises en place, les méthodes de résolutions ne sont pas désignées.

e) Une question résolue peut amener à se poser d'autres questions : il n'y a que des critères de fin locaux.

Variables de la situation

On s'intéresse ici aux différentes variables de la situation, et notamment à celles des deux types suivants :

- Les variables de recherche
- Les variables didactiques

Les *variables de recherche* sont les variables de la situation qui sont à disposition de l'élève pour organiser son travail de recherche.

> *L'élève se retrouve en effet en position de « chercheur » mais aussi en situation de « gestionnaire » de sa propre recherche : c'est lui qui choisit et modifie les valeurs des variables de recherche.*
>
> (Cartier et *al.*, 2006)

> *Elles déterminent la compréhension et l'intérêt de la question, son ouverture à de nouvelles questions, l'élargissement des stratégies de recherche, les possibilités de transformation du problème (modélisation).*
>
> (Grenier & Payan, 2003)

Les *variables didactiques* sont celles parmi les variables de la situation, dont les modifications peuvent provoquer un changement de stratégie chez l'apprenant (Brousseau, 1982). Dans la situation étudiée, nous avons repéré les variables didactiques suivantes :

Parmi les variables du problème :

- Le nombre de fausses pièces :

Le problème peut devenir relativement complexe, si l'on s'autorise plus d'une fausse pièce : par exemple le problème à deux fausses pièces n'est à ce jour pas complètement résolu pour la recherche en mathématiques. Pour le problème où l'on sait qu'il y a au maximum une fausse pièce, pour n'importe quelles valeurs des autres variables, on peut savoir si le nombre de fausses pièces est 0 ou 1 en une ou deux pesées ce qui favorisera une stratégie du type S_{PROB}.

Au contraire, si l'on sait qu'il y a exactement une fausse pièce, la question de rechercher une fausse pièce qui n'existe peut-être pas n'interfère plus et des stratégies comme S_{DICHO} ou S_{TRICHO} pourront être mises en place directement.

- Le poids des fausses pièces ou plutôt les informations sur leur poids :

Si l'on sait qu'elles sont plus lourdes (ou plus légères, c'est le même problème), ou si l'on ne le sait pas, les stratégies envisageables vont sensiblement différer. Dans le problème $P_{1,+/-\_}$ par exemple : Ne pas connaître le poids de la fausse pièce pousse à rechercher cette information avant de rechercher la fausse pièce, c'est-à-dire renforce la stratégie S_{PROB}.

Ou au contraire, on peut être amené à penser que le poids de la fausse pièce ne peut être déterminé et se tourner vers une stratégie de type S_{ETAL} , qui n'utiliserait par exemple que l'information que le poids de la fausse pièce est différent de celui d'une vraie.

Si l'on connaît à l'avance le poids de la fausse pièce ($P_{1,+}$), la première de ces stratégies n'a pas de sens. La deuxième, bien que pouvant aboutir à une méthode de recherche acceptable, sera effacée par des stratégies moins complexes et utilisant toute l'information sur le poids de la fausse pièce (S_{DICHO}, S_{TRICHO} …).

• Le nombre de pièces :

Lorsque l'on veut résoudre un cas particulier du problème avec un nombre de pièces fixé, ce nombre peut influencer les stratégies. Par exemple, un petit nombre de pièces permet la stratégie S_{EXP} mais dès que l'on s'interroge sur un grand nombre de pièces, ce type de stratégie devient impossible à mettre en œuvre (notamment une énumération de tous les cas devient très vite fastidieuse). Ou encore pour les problèmes $P_{1,+}$ ou $P_{0/1,+}$, un nombre pair de pièces poussera à partager les pièces en deux tas et à toutes les peser ensemble et favorisera S_{DICHO}. De même si l'on cherche sur un cas à $3k+1$ ou $3k+2$ pièces, S_{TRICHO} peinera à apparaître face à d'autres stratégies.

Ces trois variables, nombre de fausses pièces, poids des fausses pièces et nombre de pièces sont des variables de recherche de la situation.

Parmi les autres variables de la situation nous avons distingué deux variables didactiques :

• L'énoncé du problème : La présentation du problème et l'ordre dans lequel sont présentés [*sic*] les différentes variantes peut avoir une influence sur les méthodes de résolutions choisies pour chacun d'eux, de même que le choix des exemples préliminaires qui peuvent être proposés.

• Le matériel disponible : La présence d'une balance Roberval et de « pièces » peut avoir une influence sur la dévolution de l'énoncé du problème et sur la méthodologie de recherche. Cela peut favoriser les stratégies de type S_{EXP}, mais cela peut aussi entraver le passage à des grands nombres de pièces et créer des difficultés à énumérer tous les cas possibles pour un algorithme donné.

Hypothèses sur les stratégies

Le calcul de la complexité des algorithmes proposés sera, d'après nous, souvent interrogé. Mais il nous semble que cela aboutira plus rarement car cela met en jeu plus d'outils mathématiques. En résumé nous retenons les hypothèses suivantes :

H_1 : Pour $P_{1,+}$, la stratégie naturelle est S_{DICHO}. Les stratégies S_{EXP} et S_{TRICHO} viennent plus tard.

H_2 : L'optimalité de S_{TRICHO} sera peu abordée et lorsque ce sera le cas ce sera très majoritairement avec un argument d'optimalité locale.

H_3 : Pour les autres variantes, la stratégie naturelle est S_{PROB} pour revenir à $P_{1,+}$.

H_4 : L'optimalité sera peu abordée et lorsque ce sera le cas elle sera déduite de celle de $P_{1,+}$.

Cadre d'expérimentation

Nous avons expérimenté cette situation dans deux classes (chacune d'une vingtaine d'étudiants répartis en cinq groupes) :

– Un cours de mathématiques de la formation des PE_2, à l'Iufm de Créteil.

– Option « Jeux combinatoires et raisonnements mathématiques » proposé à l'université Joseph Fourier et ouvert aux étudiants de première et deuxième années des licences scientifiques.

Les séances se sont déroulées suivant un même plan :

1) Présentation du problème et constitution des groupes.

2) Recherche en groupe sur le problème donné. Ce moment a duré entre 1 heure et 1 heure et demi [*sic*]. Pendant tout ce temps, les gestionnaires de situation ont circulé dans les groupes pour suivre leurs avances et répondre aux questions.

3) Préparation de la présentation orale. Chaque groupe a présenté ensuite son travail à la classe (résultats, conjectures, questions en suspend...).

4) Institutionnalisation.

L'étude a porté sur les brouillons des étudiants, leurs présentations orales et des enregistrements vidéos de certains groupes en travail de recherche.

3.4 *Résultats de l'expérimentation*

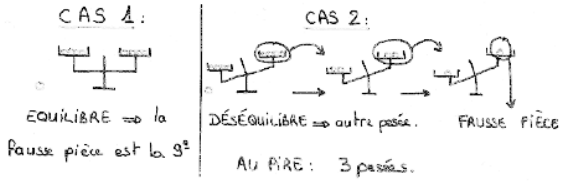
Stratégies utilisées

• Nous avons retrouvé la plupart des stratégies présentes dans l'analyse mathématique. Tous les groupes ont utilisé une représentation de la balance (plus ou moins schématique) pour symboliser une pesée.

Représentations d'une pesée.

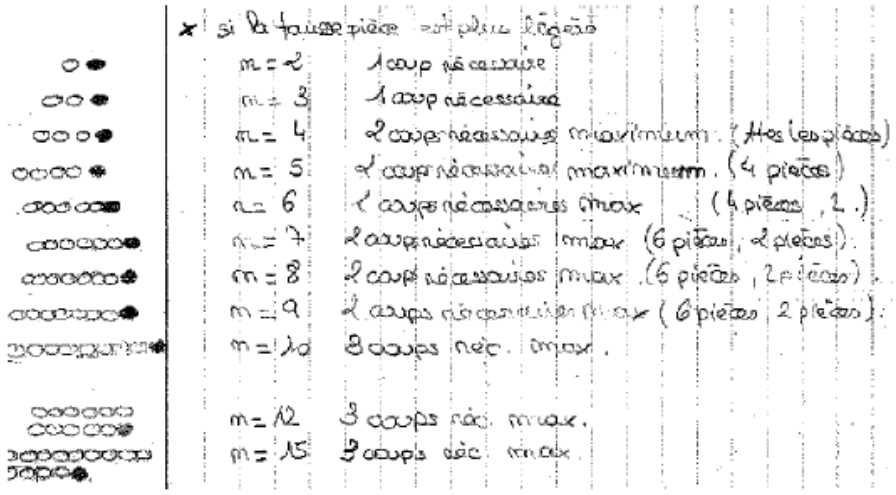
• Que ce soit pour $P_{1,+}$, $P_{1,+/-}$ ou $P_{0/1,+}$, la stratégie la plus présente est S_{DICHO} ou ses variantes. Elle est développée dans plus de la moitié des productions et évoquée dans toutes. Les groupes filmés évoquent clairement cette stratégie dès le début. Cela semble confirmer les hypothèses sur les stratégies. Cette stratégie donne souvent lieu à un algorithme qui est parfois énoncé en

langage courant, d'autres fois représenté par un schéma (dans ce cas il est généralement présenté sur un exemple).



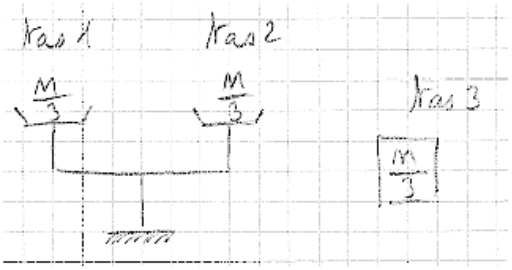
La stratégie de dichotomie.

• On retrouve aussi beaucoup la stratégie S_{EXP}, elle peut être utilisée dans deux buts : chercher la méthode optimale ou évaluer la complexité d'une méthode. On l'a clairement identifiée dans la moitié des groupes observés. Elle permet à ces groupes de valider ou d'invalider des méthodes, d'en conjecturer d'autres et de faire des hypothèses sur le nombre de pesées. On retrouve très clairement cette stratégie dans les échanges oraux, où elle est utilisée pour étudier S_{TRICHO} et conjecturer la complexité de l'algorithme construit. Les nombres de pièces plus petits sont, par exemple, utilisés pour traiter les cas suivants. Cela semble aider à repérer le lien de récurrence existant.



La stratégie expérimentale.

• Une autre stratégie que l'on retrouve souvent est la stratégie est S_{TRICHO}. On la retrouve clairement chez quatre groupes et elle semble être évoquée chez deux autres. Elle donne lieu à un algorithme, parfois difficile à préciser selon le nombre de pièces. Comme pour S_{DICHO} l'algorithme est présenté en langage courant ou par un schéma. L'optimalité de cet algorithme est souvent conjecturée mais rarement prouvée. Elle est clairement évoquée dans les discussions des groupes et apparaît dans les deux cas (parfois assez tôt).



La stratégie de trichotomie.

• S_{PROB} est aussi beaucoup utilisée chez les groupes qui ont traité plusieurs problèmes : pour résoudre $P_{1,+/-\_}$ ou $P_{0/1,+}$, ils cherchent à se ramener au problème $P_{1,+}$ qui leur semble peut-être plus simple. Sur les 7 ou 8 groupes qui ont traité plusieurs problèmes, quatre ont eu clairement recours à cette stratégie.

• On retrouve aussi des variantes de ces stratégies, par exemple mélangeant S_{TRICHO} et S_{DICHO}, ou d'autres encore, proposant des partages des pièces dans d'autres proportions. On retrouve aussi dans une production la stratégie S_{ETAL}.

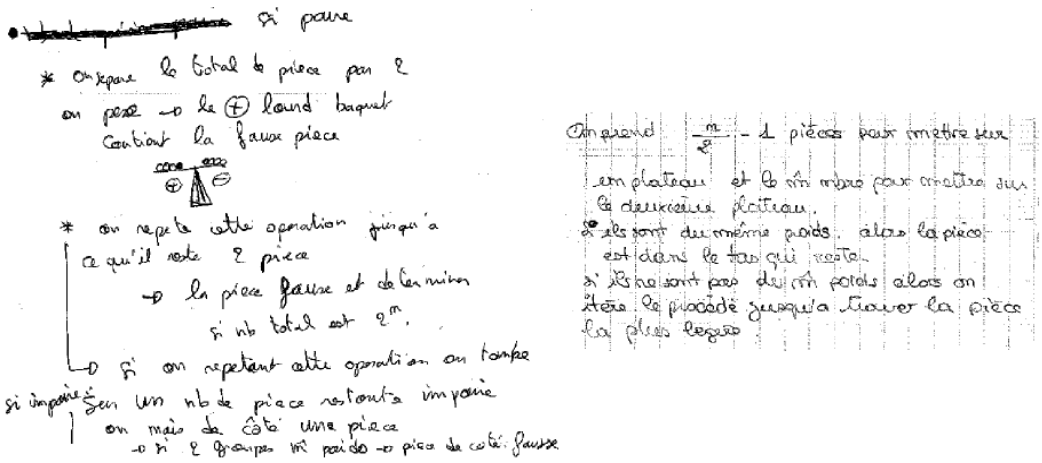
Il semble donc que la stratégie la plus naturelle soit S_{DICHO} et que S_{EXP} puisse être une bonne stratégie pour rejeter la dichotomie. Viennent alors les autres stratégies et notamment S_{TRICHO} . Il apparaît aussi que le passage par $P_{1,+}$ soit un recours naturel dans la résolution des autres variantes proposées. La majorité des stratégies évoquées dans l'analyse mathématique sont présentes et leur ordre prévu dans l'analyse a priori semble se confirmer.

Algorithmes

• Les étudiants produisent des algorithmes sous deux formes principalement : en langage courant et sous forme de schémas (arbres ou succession de pesées). Dans le deuxième cas l'algorithme est souvent présenté sur un exemple. Il est alors difficile de savoir si l'algorithme a été généralisé à un nombre quelconque de pièces ou non. La construction d'algorithmes apparaît très nettement dans les échanges oraux à divers moments. Les descriptions sont très claires et l'on retrouve des termes tels que « la même procédure sera appliquée » ou « jusqu'à ce que tu tombes sur plus qu'une pièce » qui mettent en évidence une itération. Les algorithmes construits sont tous des algorithmes valides de recherche de la fausse pièce, il semble que la question de produire un ou plusieurs algorithmes pour le problème ne pose pas de grosses difficultés.

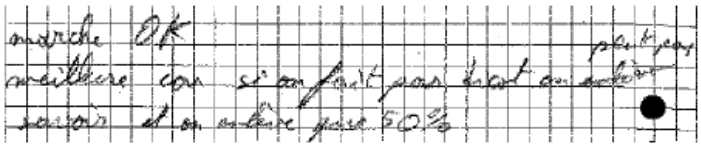
• Se pose la question de la preuve de ces algorithmes. Elle transparaît très peu dans les documents écrits des étudiants mais on peut constater que dans la majorité des productions, tous les cas possibles pour la fausse pièce sont pris en compte. On constate aussi l'utilisation (en apparence) d'invariants pour traiter moins de cas (symétrie, cas similaires, retour à un nombre

de pièces déjà traité...). Concernant la preuve des algorithmes, l'étude des échanges oraux nous en dit un peu plus. En effet, la preuve des algorithmes transparaît à de nombreux endroits et de manière très nette. Elle a lieu en même temps que la construction des algorithmes, par disjonction des différents cas : « on a soit *2k* pièces soit *2k+1* pièces » ou « Si c'est ni dans l'un ni dans l'autre... » par exemple.



Deux algorithmes en langage courant.

• Les algorithmes sont parfois comparés entre eux, sur des exemples ou sur un nombre quelconque de pièces, ce qui permet de rejeter, pour un algorithme trouvé, la conjecture selon laquelle il est optimal. La stratégie S_{EXP} développée par plusieurs groupes peut aussi relever de la comparaison d'algorithmes. Les comparaisons d'algorithmes ne sont faites que sur des exemples ou par des arguments d'efficacité locale (sur une pesée). On voit clairement ces comparaisons dans les échanges des étudiants.



Comparaison des stratégies de trichotomie et dichotomie.

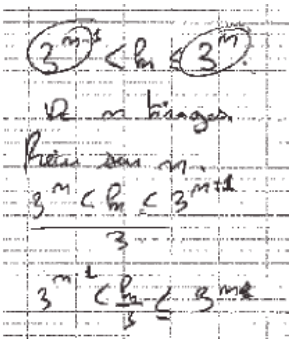
• La complexité, ou plutôt le nombre de pesées, est questionnée par plusieurs groupes, souvent sans succès. Cependant quelques groupes émettent des hypothèses sur ce nombre de pesées, souvent à l'aide de S_{EXP}. Certains vont jusqu'à produire une preuve de la complexité de leur méthode que nous préciserons dans la partie concernant la preuve. Dans les échanges, la complexité est questionnée et l'on retrouve des calculs (ou des tentatives de calcul) de cette complexité. Par exemple, « On peut calculer le nombre de pesées mais ça dépend de *n* », évoque clairement la complexité. L'optimalité est aussi questionnée et ce sont plutôt les échangent oraux qui nous en informent. De manière générale, il semble que l'enjeu d'optimisation ait vécu

dans l'activité des groupes. Le critère de complexité au pire a été sujet à discussion dans les groupes. Les questions d'optimalité ont aussi donné lieu, par exemple, à des discussions sur le passage d'un problème à un autre et la conservation de l'optimalité, qui sont des questions importantes de l'algorithmique.

Démarche de preuve

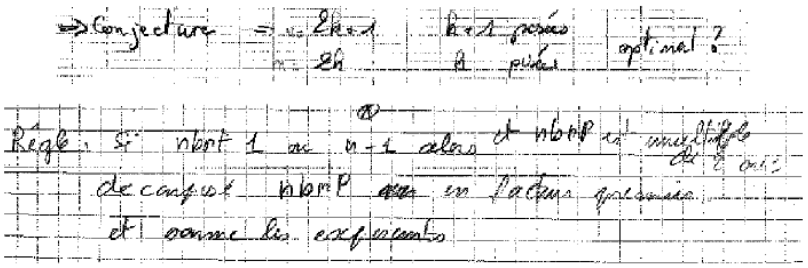
D'après les résultats présentés jusqu'ici, un enjeu de preuve semble être présent dans l'activité des étudiants. Elles peuvent être des preuves d'algorithmes, des preuves de complexité ou des preuves d'optimalité. Voyons plus précisément quels sont les éléments constitutifs de cette activité.

• D'une part, quelques preuves sont présentes dans les productions écrites, la difficulté est qu'elles ne sont pas toujours très formalisées et il est difficile de les identifier clairement. L'étude des échanges oraux nous a permis de mieux repérer ces preuves, nous les avons déjà évoquées dans les paragraphes précédents.



Une preuve par récurrence.

• D'autre part, des hypothèses, des conjectures et des questions sont présentes sur les productions. Elles sont significatives d'une démarche de preuve, mais sont, pour beaucoup, formulées oralement. Citons comme exemples : « Ah ouais, c'est les plus équilibrés en fait, je pense... », « Mais 5 tu l'auras pas avant 90. » ou « Mais pourtant c'est la meilleure méthode. Enfin on suppose que. »...



Exemples de conjectures.

• On retrouve aussi l'expression d'une nécessité de recourir à la preuve dans les discussions, par exemple dans un groupe : « Je pense quand même qu'il y a une raison un peu plus... logique », « Ben allez, faut le prouver. », « Une récurrence pour prouver quoi ? » ou « Si il te la prouve pas comment tu vas le croire ? »... Ou dans un autre groupe : « Je pense plutôt qu'on est pas assez rigoureux. » ou « On va faire un raisonnement en probabilités... ».

• L'utilisation de contre-exemples permet aussi de déceler une activité de preuve. Cependant, il est difficile de voir sur les productions écrites si des contre-exemples sont utilisés. On peut en repérer mais ils sont sûrement plus souvent évoqués à l'oral. Les comparaisons de méthodes entre elles constituent aussi des contre-exemples à l'optimalité de l'une des deux. S_{EXP} permet de construire de nombreux exemples qui peuvent servir de contre-exemples à l'optimalité de certaines méthodes. C'est ce que semble faire l'un des groupes : « Si tu fais *4*, *4* et *2*. Si t'enlèves les *2*, en gros t'auras, t'auras perdu une pesée quoi. » pour rejeter une stratégie de découpage en *2*, *2*, *1*.

Comparaison des deux publics

Même si nous avons présenté ensembles les résultats des deux expérimentations, il est nécessaire de discuter des différences et des points communs entre les deux publics observés.

• Concernant les stratégies mises en œuvre, on rencontre chez les deux publics les stratégies S_{DICHO} et S_{EXP} de façon courante. La stratégie S_{PROB} est aussi commune aux deux expérimentations. Cependant, les PE_2 n'ont pas soulevé S_{TRICHO} que l'on ne retrouve que chez les étudiants de licences scientifiques.

• Concernant les algorithmes, leur construction est mise en œuvre par les deux publics mais le passage au cas général semble plus difficile pour les PE_2 et tous ne l'ont pas traité. Comme la preuve de ces algorithmes se fait généralement en parallèle de leur construction, on la retrouve aussi chez les deux publics. L'enjeu d'optimisation semble bien présent dans les deux expérimentations et la notion de complexité au pire paraît assez naturelle pour tous les étudiants. Des tentatives de calcul de cette complexité existent aussi bien dans les deux expérimentations, mais il semble qu'elles n'aboutissent pas chez les étudiants de PE_2. Concernant l'enjeu de preuve, on le retrouve à des degrés différents dans les deux cas. Cependant des preuves d'optimalité ne sont présentes que chez les étudiants de L_1 et L_2.

En conclusion

On constate, sans surprise, que le rapport à la preuve des deux publics auprès desquels nous avons expérimenté est différent. Ces résultats, concernant la mise en place de preuves semblent

aller dans le sens de ceux de Schuster qui a étudié l'introduction de problèmes d'optimisation combinatoire dans l'enseignement secondaire (Schuster, 2004). Il constate que la construction d'algorithmes peut vivre en classe dès le grade 9 et que la preuve n'est réellement mise en œuvre qu'aux grades 11 et 12 de manière autonome [Note 14. Le grade 9 correspond à la tranche d'âge 15-16 ans et les grades 11 et 12 à la tranche 17-19 ans.] Il serait pertinent d'étudier le rapport à la preuve des différents publics concernés par cette situation.

Cette situation a permis à certains groupes de mettre en œuvre la construction d'algorithmes et de questionner certains aspects de l'algorithme (preuve, complexité, optimalité) qui sont absent [*sic*] des programmes et manuels. Ces aspects sont du côté objet de l'algorithme. Concernant les hypothèses sur les stratégies, elles se sont vérifiées ici. La construction et la preuve d'algorithmes ont été effectuées par une majorité d'étudiants. La question de l'optimalité a été rarement résolue et les groupes qui l'ont traité ont privilégié l'optimalité locale. Pour beaucoup d'étudiants, traiter l'optimalité demanderait sûrement un temps plus long.

Cette situation semble donc porteuse d'un potentiel pour la manipulation de l'algorithme en tant qu'objet. Il semble donc important de poursuivre son étude sur une période plus longue (5 à 6 séances de 2 heures par exemple pour une SiRC classique). Cette étude pourrait être mise en œuvre avec différents publics (du primaire à l'université et en formation d'enseignants) afin d'étudier le rapport de chacun à l'algorithme et de comprendre l'évolution de ce rapport au fil de la situation. (pp. 60-71)

Éléments d'analyse 3.2

1. La partie 3, « Une situation pour l'algorithme », est, nous l'avons indiqué, la plus longue des trois parties composant l'article examiné (elle occupe les pages 60 à 71). Cette partie a pour objet de montrer « que l'on peut proposer des situations permettant l'accès aux différents aspects de l'algorithme et notamment à l'aspect objet ». On a là, sous-jacent, un principe qui est évidemment essentiel au MDR de courants importants de la didactique des mathématiques : *l'organisation d'un cursus d'études se construit à l'aide de situations qui permettent de provoquer une rencontre pensée comme décisive – au plan didactique – des élèves ou étudiants avec des connaissances regardées comme cruciales dans la formation suivie et souvent emblématique de la discipline étudiée* ⟨9⟩. Ce principe va de pair avec un autre principe qu'on précisera peu à peu : *une situation, au sens usité par les auteurs, a pour noyau (ou cœur) un certain problème mathématique* ⟨10⟩. Les auteurs annoncent d'emblée qu'il s'agit en l'espèce d'un « problème de recherche de fausses pièces ».

① C'est ici l'occasion de rappeler brièvement ce qu'est, d'une manière générale, un *problème*. Voici la notice que consacre au mot *problem* (en anglais) le dictionnaire de John Ayto, *Dictionary of Word Origins* (1990) :

> **problem** [14] A *problem* is etymologically something 'thrown forward.' The word comes via Old French *probleme* and Latin *problēma* from Greek *próblēma*, a derivative of *probállein* 'throw forward.' This was a compound verb formed from the prefix *pro-* 'forward' and *bállein* 'throw' (source of English *ballistic, emblem, parable,* etc.). Things that are 'thrown out' project and can get in the way and hinder one, and so *próblēma* came to be used for an 'obstacle' or 'problem' – senses carried through into English *problem*.
> ► baIlistic, emblem, parable, symbol

Un problème, c'est donc un défi que l'on lance devant un collectif (une assemblée, une classe, etc.). Ce collectif se trouve alors confronté à un obstacle, qu'il doit surmonter en « résolvant le problème ».

② Le problème proposé est « un problème de fausses pièces ». Il est plus fréquent, dans les initiations à l'algorithmique, de proposer des « problèmes de tri », un domaine bien travaillé à la fois du point de vue « savant » et du point de vue de l'enseignement. Voici par exemple comment le mathématicien Pierre Damphousse présentait ce type de recours dans son ouvrage *Petite introduction à l'algorithmique. À la découverte des mathématiques du pas à pas* (Ellipses, 2005) :

> **1. Les algorithmes de tri**
> *1.1. Introduction* – Ce paragraphe présente les algorithmes de base formant le minimum de toutes les formations en informatique, dans lesquelles ils offrent l'occasion de bons exercices de programmation. Ils offrent aussi l'occasion de découvrir les tâches basiques de l'algorithmique : la *mesure de la complexité* et la *démonstration de la convergence.* Pour certains d'entre eux, ils permettent en plus de se familiariser avec la *récursivité.* On étudiera ici le tri par insertion, le tri par fusion, le tri rapide, la tri par tas, et des algorithmes de tris non comparatifs. C'est en se familiarisant avec ce premier jeu d'algorithmes que le lecteur prendra progressivement la mesure des notions et des objectifs de base de l'algorithmique.
> Ce paragraphe n'est qu'un très mince échantillonnage de l'ensemble des algorithmes de tri; Dans [9], plus d'une trentaine d'algorithmes sont analysés avec finesse, et il faut ajouter à ces

> algorithmes bon nombre d'autres dont l'étude est donnée en exercice. Notre objectif sera atteint si le lecteur, même avec un mince bagage mathématique, entrevoit la richesse du sujet et y découvre les questions de base de l'algorithmique. (p. 9)

La référence [9] de l'auteur est en fait la « bible de l'algorithmique et de la programmation », à savoir *The Art of Computer Programming* en trois volumes (1968, 1969, 1973) de Donald E. Knuth. On peut se demander pourquoi les auteurs ont choisi, non ce type de problèmes, mais celui des fausses pièces. Il n'est pas entièrement déraisonnable d'imaginer que si, à l'instar de qui aurait à enseigner de l'algorithmique à des étudiants débutant en la matière, ils avaient « banalement » choisi comme moteur mathématique ce grand classique que sont les problèmes de tri, ils n'auraient pu regarder et *faire regarder* leur travail comme suffisamment novateur pour être qualifié de travail de *recherche*. On pourrait ainsi identifier un complément aux deux principes précédemment énoncés : *une organisation didactique en mathématiques vaut moins par les entités praxéologiques qu'elle vise à faire rencontrer que par le problème mathématique choisi pour engendrer la rencontre avec ces entités* ⟨11⟩.

③ Les problèmes de fausses pièces sont à la fois des classiques dans le registre des jeux d'énigmes (voir par exemple l'article « Balance Puzzle » de *Wikipedia*) et des problèmes mathématiques dont certains sont presque triviaux et d'autres restent *ouverts*. Dans leur article « Coin-Weighing Problems » paru dans le numéro 102 (février 1995) de l'*American Mathematical Monthly*, pp. 164-167, les auteurs, Richard K. Guy et Richard J. Nowakowski commencent leur exposé par ces lignes :

> The question of finding a single counterfeit coin from a set of regular coins in the fewest number of weighings using just a balance beam has been a notorious problem. The regular coins are all the same weight while the counterfeit coin is a different weight.

Présentant alors un historique des travaux, ils notent d'abord ceci, qui souligne que ces problèmes peuvent devenir obsédants :

> The problem was popular on both sides of the Atlantic during World War II ([14, 15, 20, 21, 27, 34, 39, 461; it was even suggested that it should be dropped over Germany in an attempt to sabotage their war effort; see [35, 40, 43] for some history.

Guy et Nowakowski poursuivent alors par un bref bilan de résultats obtenus à la date de la rédaction de leur article (1995) :

> In solving [39] Kaplansky, Neugebauer and Pennell gave the following general solution to the problem of underweight counterfeit coins: If $3^{n-1} \leq N < 3^n$ *then n weighings suffice to show if there is (and to identify) a counterfeit coin among N coins*. If it is known that a counterfeit coin exists then n weighings will identify the coin from among N coins if $3^{n-1} < N \leq 3^n$. Dyson [12] gave an elegant solution using ternary labels when it is not known if the counterfeit coin is heavy or light; see [11] for a solution in verse. In this case, n weighings suffice
>
> (i) if $N \leq (3^n - 3)/2$ and it is required to find if the dud is heavy or light;
>
> (ii) if $N \leq (3^n - 1)/2$, given an extra coin known to be good, and it is required to find if the dud is heavy or light; and
>
> (iii) $N \leq (3^n + 1)/2$ if there is a good coin but the relative weight of the counterfeit coin is not required.

Le lecteur intéressé (notamment par les références bibliographiques) pourra trouver en ligne l'article cité (http://www.mathstat.dal.ca/~rjn/MATH4370/attachments/GuyN1995b.pdf). Sans se laisser obséder par le problème des fausses pièces, on trouvera des développements le concernant à l'adresse http://www.cut-the-knot.org/blue/12CoinsInVerse.shtml#solution. Nous n'irons pas plus loin sur ce problème ; nous nous efforcerons d'envisager ce qu'on peut induire, quant à un MDR possible, de l'emploi qu'en font les auteurs.

2. Le problème de base que retiennent les auteurs est énoncé ainsi :

> Dans un ensemble de pièces, indiscernables à la vue ou au toucher, se trouvent des fausses pièces. Les vraies pièces pèsent toutes le même poids, les fausses aussi mais leur poids est différent de celui des vraies. A l'aide d'une balance Roberval à deux plateaux et sans poids, comment peut-on retrouver les fausses pièces ? Quelle est la méthode qui permet de les retrouver en effectuant le moins possible de pesées ?

Mais ils considèrent plus précisément des « assortiments » de variables qui paramètrent le problème :

> On peut définir plusieurs variantes de ce problème en jouant sur les variables de la situation suivantes :

- Poids des fausses pièces :

– On sait au départ que les fausses pièces sont plus lourdes que les vraies (ou plus légères).

– On ne sait rien.

- Le nombre de fausses pièces :

– Fixé à l'avance.

– Compris dans un intervalle donné.

– Non connu à l'avance.

① Les auteurs notent que la plupart des variantes du problème des fausses pièces « ne sont pas encore complètement résolus pour la recherche en mathématiques ». Ils ajoutent que les variantes effectivement mises au cœur de la « situation pour l'algorithme » qu'ils ont étudiée étaient censées « permettre une mise en œuvre d'algorithmes et leur étude par des élèves », condition qu'ils expliciteront un peu plus loin.

② On a là un type de situation non classique dans l'enseignement secondaire, où, d'ordinaire, si les problèmes proposés aux élèves sont, certes, non encore résolus, ils ne le sont que *dans la classe seulement*. À titre d'éclairage, voici un extrait du résumé proposé par son auteur, Ratko Tošić (1983), d'un article paru dans *Discrete Mathematics* (46(3), 295-298) sous le titre « Two Counterfeit Coins », article auquel renvoient les auteurs que nous suivons (voir à l'adresse http://www.sciencedirect.com/science/article/pii/0012365X83901231) :

> We consider the problem of ascertaining the minimum number of weighings which suffice to determine the counterfeit (heavier) coins in a set of *n* coins of the same appearance, given a balance scale and the information that there are exactly two heavier coins present. An optimal procedure is constructed for infinitely many *n*'s, and for all other *n*'s a lower bound and an upper bound for the maximum number of steps of an optimal procedure are determined which differ by just one unit.

Il est vrai que le problème $P_{1,+}$ considéré par les auteurs – problème où il n'y a qu'*une* fausse pièce supposée *plus lourde* que les pièces authentiques – a une solution « bien connue », comme on le verra plus loin. Mais cette solution ne semble pas immédiatement accessible au profane. À cet égard, on peut conjecturer que le MDR des auteurs pourrait inclure un principe semblable à celui-ci : *pour mettre une situation didactique à l'abri de perturbations indésirables par l'environnement ordinaire de la classe, il est bon que le problème au cœur de la situation soit, dans sa généralité, et sans aide documentaire ou autre, hors de portée des*

élèves ou étudiants, par exemple parce qu'il s'agit d'un problème encore mathématiquement ouvert ⟨12⟩.

.

3. Les auteurs explicitent ensuite ce que leur semble être la présence de « l'algorithmique dans ce problème ». Celui-ci, affirment-ils, « relève bien d'une problématique d'algorithmique ». Pourquoi cela ? Parce que, pour qui s'y confronte, « il s'agit de construire un algorithme de recherche des fausses pièces, de le prouver et d'étudier son optimalité ».

① Selon les auteurs, la réception du problème par les élèves ne ferait pas problème : « L'énoncé du problème ne fait appel à aucune notion mathématique particulière mise [*sic*] à part la notion d'optimalité, mais qui peut-être comprise au sens intuitif. L'appropriation de ce problème nous paraît donc relativement facile. » Une fois cette « appropriation » consommée, la construction d'un algorithme semblerait pouvoir être amorcée à peu de frais : le fait de s'assurer que tel algorithme conduit bien au résultat attendu – l'identification de *la* fausse pièce – relèverait d'une logique « intuitive » ; quant à l'optimalité, l'idée qu'un algorithme est plus « efficace » localement (sur des cas particuliers) qu'un autre – qu'il nécessite moins de pesées – semblerait évidente. Mais les auteurs arrivent là à un point de difficulté.

② Ce qui semble « naturel », « intuitif », c'est « la complexité dans le pire des cas ». Mais, ainsi que nous l'avons vu, il est au moins une autre notion de complexité : la complexité *en moyenne*. Supposons par exemple que l'on ait $n = 12$ pièces. On adopte la technique suivante : on pèse l'une contre l'autre les deux premières pièces c_1 et c_2 ; s'il y a déséquilibre, une pesée suffit pour conclure ; sinon, on compare les deux pièces suivantes, c_3 et c_4, etc. Si la fausse pièce est la pièce c_{2p-1} ou c_{2p}, où $p = 1, \ldots, 6$, elle sera découverte en p pesées. Dans le pire des cas, il faudra 6 pesées pour parvenir au but. La moyenne des pesées est donc la moyenne du caractère $c_m \mapsto \lceil m/2 \rceil$, qui vaut $(2 \times 1 + 2 \times 2 + 2 \times 3 + 2 \times 4 + 2 \times 5 + 2 \times 6)/12 = 3{,}5$.

③ Cette notion, construite et présente dans l'algorithmique « savante », pèse sur le travail des auteurs. Car, contrairement à la complexité au pire, elle a moins de chances d'apparaître par génération spontanée dans le travail des élèves. Les auteurs l'écartent donc de l'article examiné : « Ici, notent-ils *in fine*, nous ne nous intéresserons pas au problème de l'optimalité pour la complexité en moyenne. » Sans doute gênés par cette mise à l'écart, ils se défendent d'une accusation que d'aucuns – des « spécialistes » ? – pourraient peut-être porter contre eux : « Cela ne veut pas dire, soulignent-ils, que cette question n'est pas intéressante du point de vue mathématique ou didactique. »

④ Ce qui précède conduit à expliciter un nouveau principe : *le problème à étudier agit de lui-même, « naturellement », sur les élèves, dont il provoque spontanément la rencontre avec les notions visées* ⟨13⟩. Quand telle notion n'est pas, « en moyenne », ainsi rencontrée, il suffit de… l'écarter du nombre des notions auxquelles on « s'intéresse ». Il y a en cela la marque d'un certain *naturalisme didactique* dont on retrouvera ailleurs d'autres manifestations.

4. Les auteurs proposent alors une « étude mathématique du problème ». Il s'agit là d'une étape clé – qui participe de « l'analyse a priori » – dans le travail sur une situation organisée autour d'un problème déterminé : rechercher les « stratégies » possibles d'attaque du problème. Pour ce qui est du problème $P_{1,+}$ les auteurs considèrent cinq stratégies (qui sont autant de *techniques* en cours d'élaboration) : la « stratégie expérimentale » (S_{EXP}) ; la « stratégie de dichotomie » (S_{DICHO}) ; la « stratégie de trichotomie » (S_{TRICHO}) ; la « stratégie par étalon » (S_{ETAL}) ; enfin la « stratégie de passage par un autre problème » (S_{PROB}). Comme souvent dans ce genre d'exercice, on ignore ce qui a déterminé le choix des auteurs : on imagine que les stratégies présentées ne sont pas regardées par les auteurs comme les seules possibles, puisque d'autres techniques existent de manière obvie (telle la technique de comparaison par paires explicitée plus haut dans le cas $n = 12$).

① L'absence de mise en débat de cette analyse du problème renforce l'impression « naturaliste » déjà évoqué : l'espace du travail (d'une classe) sur le problème des fausses pièces serait structuré comme il a été indiqué et, craint-on de devoir comprendre, pas autrement.

② Les auteurs mentionnent un résultat mathématique relatif à la stratégie S_{TRICHO} :

- Stratégie de trichotomie S_{TRICHO} :

  On divise le tas de pièces restantes en trois paquets de « même taille » i.e. tels que

  $$|A| = |B| \text{ et } | |A|-|C| | \leq 1.$$

  Pour n pièces, avec $3^{k-1} < n \leq 3^k$, cet algorithme demande k pesées, autrement dit $[log_3(n)]$ pesées ou encore cette méthode permet de traiter en k pesées jusqu'à 3^k pièces.

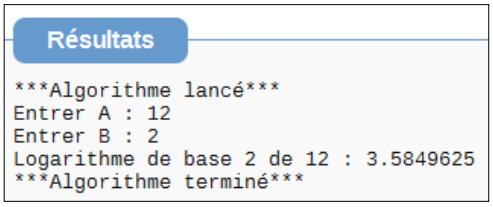
  Proposition. Cet algorithme est optimal pour la complexité au pire.

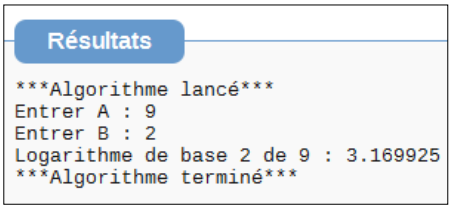
❶ Notons une erreur dans les lignes précédentes : on a $k = \lceil \log_3 n \rceil$ et non $k = \lfloor \log_3 n \rfloor$. La démonstration de cette proposition (correctement formulée) se trouve dans la thèse de Simon Modeste, pp. 220-221, qui en donne « 3 preuves ».

❷ Il semble que cette stratégie optimale apparaisse moins spontanément que la technique de dichotomie, à propos de laquelle les auteurs écrivent ceci :

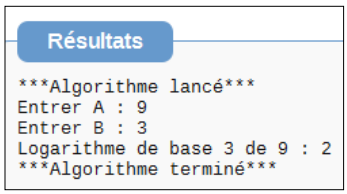
> Pour traiter *n* pièces, on compare « la moitié » des pièces sur chaque plateau. Le déséquilibre permet de garder *[n/2]* pièces à tester [Note 12. On notera entre crochets *[ ]* la partie entière inférieure.], on réitère ensuite le processus. Si *n* est impair, un équilibre de la balance est possible et indique que la fausse pièce est celle laissée de côté. Cela donne un algorithme en $[log_2(n)]$ pesées au pire.

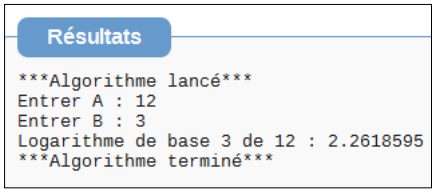
Soit k tel que $2^k \leq n < 2^{k+1}$. On a en général $2^{k-m} \leq n/2^m < 2^{k-m+1}$ et en particulier $1 \leq n/2^k < 2$. *En supposant le pire des cas*, on effectuera une première pesée pour comparer deux lots de $\lfloor n/2 \rfloor$ (= 2^{k-1}) pièces, une deuxième pesée pour comparer deux lots de $\lfloor n/2^2 \rfloor$ (= 2^{k-2}) pièces, une troisième pesée pour comparer deux lots de $\lfloor n/2^3 \rfloor$ (= 2^{k-3}) pièces, …, enfin une k-ième pesée pour comparer deux lots de $\lfloor n/2^k \rfloor$ (= $2^0 = 1$) pièce, c'est-à-dire deux lots d'une pièce chacun. Il faudra ainsi en tout, dans le pire des cas, k pesées. En prenant l'image par la fonction $\log_2$ de l'encadrement $2^k \leq n < 2^{k+1}$, on obtient l'encadrement $k \leq \log_2 n < k+1$, en sorte que $k = \lfloor \log_2 n \rfloor$. Prenons par exemple $n = 12$; on a $\log_2 12 = 3{,}58\ldots$ et donc $\lfloor \log_2 12 \rfloor = 3$. Pour $n = 9$, on a $\log_2 3 = 3{,}16\ldots$ et donc aussi $\lfloor \log_2 9 \rfloor = 3$.





❸ Les auteurs font observer que la stratégie par dichotomie n'est pas optimale : pour $n = 9$, la stratégie par *trichotomie* demande $\lceil \log_3 9 \rceil$ pesées, soit 2 pesées (voir ci-après). De fait, si on divise le lot de 9 pièces en trois lots de 3, une première pesée (comparant deux de ces lots) permet de déterminer celui des trois lots qui contient la fausse pièce ; et une seconde pesée entre deux des trois « lots » d'*une* pièce qu'on est alors amené à considérer permet de conclure. Pour $n = 12$, en revanche, on a $\lceil \log_3 12 \rceil = \lceil 2{,}26\ldots \rceil = 3$: trois pesées sont donc nécessaires.





③ Il appert de tout cela que le MDR conjecturé est compatible avec ce principe : *l'organisation didactique d'une rencontre censément impulsée par un problème suppose une étude* préalable – *« a priori » – assez poussée de ce problème* ⟨14⟩. Le fait d'avoir étudié *à l'avance* ce problème permet de prévoir par exemple que si des élèves prétendent traiter le problème à 9 pièces par la stratégie « dichotomique », leur traitement ne sera pas optimal, etc.

④ Notons ici que, par contraste, en TAD, dans la *pédagogie de l'enquête*, une telle analyse *a priori* par l'enseignant tend à être remplacée par une analyse *in vivo*, par la classe sous la direction de l'enseignant. C'est – en principe – ensemble qu'on découvrira (ou qu'on redécouvrira, pour ce qui est de « l'enseignant ») que, comme le notent à peu près Guy et Nowakowski (1995), « si $3^{k-1} < n \leq 3^k$, k pesées suffisent pour identifier la fausse pièce parmi n pièces ».)

5. Les auteurs donnent à la suite de la section 3 de l'article (pp. 64-71) le découpage suivant :

3.3 *Analyse de la situation*
Une Situation de Recherche en Classe
Variables de la situation
Hypothèses sur les stratégies
Cadre d'expérimentation
3.4 *Résultats de l'expérimentation*
Stratégies utilisées
Algorithmes
Démarche de preuve
Comparaison des deux publics
En conclusion

► ***Nous laisserons le lecteur mener à bien l'analyse des développements correspondants.***