PLANIFICATION ET CONNAISSANCES MATHÉMATIQUES DANS UNE SITUATION D'APPRENTISSAGE AU LYCÉE : L'ALGORITHME DE KAPREKAR

Jean-Baptiste LAGRANGE
Université de Reims et LDAR Univ. Paris-Diderot

Marie-Noëlle GUY
Cité scolaire François Villon, Paris

Résumé. Cet article étudie dans le contexte de l'algorithmique en mathématiques au lycée, la conception d'une situation d'apprentissage basée sur l'élaboration d'un algorithme mettant en jeu la « planification » ainsi que des connaissances dans un champ des mathématiques scolaires. Le cadre théorique fournit des outils relatifs à l'activité de débutants en programmation, et aux situations d'apprentissage. L'ingénierie didactique est basée sur une tâche d'expression en langage algorithmique d'un traitement « manuel », l'« algorithme » de Kaprekar. Elle est préparée par une étude de ressources existantes et met en jeu les données et leur représentation, par un questionnement enclenchant la planification. Elle montre l'intérêt, dans le contexte du curriculum du lycée, d'une approche fondée sur la psychologie de la programmation comme cadre cognitif et sur la théorie des situations didactiques comme cadre didactique.

Mots-clés. Algorithmique, lycée, Kaprekar, psychologie de la programmation, théorie des situations didactiques.

Abstract.This article studies the design of a learning situation within high school mathematics teaching/learning, based on the development of an algorithm putting at stake "planification" as well as knowledge in a field of school mathematics. The theoretical framework provides tools related to the beginners' programming activity, and to learning situations. The didactical engineering is based on a task of expressing a "manual" treatment, the Kaprekar "algorithm", in an algorithmic language, It is prepared by a study of existing resources. The situation involves data and their representation in an initial questioning in order to trigger planification. It shows the interest, in the context of the high school curriculum, of an approach based on the psychology of programming as a cognitive framework and on the theory of didactical situations as a didactical framework.

Key words. Algorithmics, high school mathematics, Kaprekar, psychology of programming, theory of didactical situations.

1. Introduction

L'introduction de l'algorithmique en 2009 dans le curriculum de mathématiques au lycée fait suite à des tentatives antérieures, non suivies d'effet dans les classes, d'instituer des tâches en algorithmique dans cet enseignement (Lagrange 2002). Tout comme les récentes propositions d'initiation au « codage informatique » à tous les niveaux de l'enseignement scolaire[1], cette introduction réactualise des problématiques qui étaient apparues dans les années 80 avec l'option informatique des lycées où la construction d'algorithmes était proposée comme une activité scientifique (Baron, 1989). Cette introduction a donné lieu à la publication de ressources pour les enseignants, souvent produites par les enseignants eux-mêmes. Celles-ci peuvent constituer des propositions de situations de classe, motivées par des objectifs d'enseignement. En revanche, il existe à notre connaissance très peu d'études relatives aux élèves, à leurs difficultés et à leurs succès lorsqu'ils sont confrontés à de telles tâches. Ceci a attiré notre attention car

1 « L'éducation au numérique, ce sera l'initiation au codage informatique dès l'école primaire. » Benoît Hamon, ministre de l'Éducation nationale, Assemblée nationale, 15 juillet 2014

l'option informatique des lycées avait, quant à elle, donné lieu à quelques études didactiques portant sur les élèves. Celles-ci, comme nous allons le voir, montraient que beaucoup parmi eux rencontraient de grandes difficultés à construire seuls des algorithmes, même simples.

Ces difficultés concernaient deux aspects : (1) la compréhension des contraintes d'un langage algorithmique et (2) le découpage en étapes et leur organisation lors de la construction d'algorithmes[2]. Selon nous, ce sont des difficultés auxquelles, sauf exception, les débutants en programmation sont confrontés, et des observations informelles nous ont confirmé qu'elles existaient bien dans le cas de l'algorithmique au lycée. Les difficultés relatives à l'aspect (1) ont été documentées et analysées notamment par Samurçay (1985) et Lagrange (1992a, 1992b), et des pistes de remédiation ont été proposées. L'analyse insiste sur la distance entre les contraintes d'expression dans un langage algorithmique et dans le langage habituel, et les pistes de remédiation visent à faire prendre conscience aux élèves de cette distance. Notons aussi que des études, notamment aux Etats-Unis, ont portées sur des activités de programmation par des élèves plus jeunes ; ces études étaient orientées vers l'évaluation de la contribution de ces activités à des compétences générales, notamment la « pensée procédurale ». Comme le souligne Crahay (1987) dans une revue de travaux autour de LOGO, les études empiriques ont montré des résultats décevants. L'auteur situe notamment les difficultés dans une approche naïve d'un « apprentissage sans instruction, (...) les enseignants (s'étant) imaginés que leur rôle dans l'environnement LOGO était minime. » (ibid p, 46).

Nous nous sommes donc étonnés de ce que les ressources produites à l'occasion de l'introduction de l'algorithmique ne s'intéressent pas davantage aux élèves, à leurs difficultés et aux stratégies enseignantes pour faire face aux problèmes rencontrés. Dans cet article, nous nous centrons sur le second aspect distingué plus haut. En effet, même pour des tâches simples, le découpage en étapes et leur organisation dans un algorithme peuvent ne pas être triviaux et imposer une prise de distance avec les traitements « manuels » familiers aux élèves, analogue à celle qui est nécessaire pour la compréhension du langage (Samurçay et Rouchier 1985). Comment les élèves du lycée peuvent-ils être confrontés à cette prise de distance ? Comment peuvent-ils progresser ?

Le curriculum de mathématiques du lycée fournit un contexte pour reposer ces questions, mais aussi pour les aborder dans une perspective différente. En effet, il est précisé dans les objectifs que : « l'algorithmique a une place naturelle dans tous les champs des mathématiques et les problèmes posés doivent être en relation avec les autres parties du curriculum (analyse, géométrie, statistiques et probabilités, logique) » alors que l'option informatique des lycées ne liait pas les tâches en algorithmique à une discipline scolaire ; il s'agissait «de développer des aptitudes intellectuelles, la créativité, pour l'invention de nouveaux algorithmes, la rigueur de pensée, pour que ces algorithmes résolvent le problème posé » (Arsac 1988, p.79). Ainsi, pour le curriculum de mathématiques du lycée, l'activité algorithmique des élèves est orientée, non plus vers

2 Pour nous, un algorithme est l'expression d'un traitement à l'aide d'un système de notation formel qui permet de l'étudier comme un objet abstrait. Nous suivons en cela Dijksra (1976, p 7) : « some suitable formal notation (is needed) to study algorithms as mathematical objects ». Dijksra précise (p. 8) que la description formelle permet l'exécution sans intervention humaine et que le système de notation est ainsi un « langage de programmation », mais que, pour lui, un tel langage est d'abord un véhicule pour la description de mécanismes abstraits. Nous employons dans cet article le terme « langage algorithmique » pour désigner un système formel permettant de produire un texte qui soit aussi bien le support d'une réflexion abstraite et qu'un objet à faire exécuter par un dispositif ad hoc.

des aptitudes générales, mais vers des compétences ou connaissances mathématiques dans les champs classiques des mathématiques scolaires.

Notre problématique générale peut par conséquent s'énoncer comme suit : est-il possible de concevoir une situation d'apprentissage imposant un découpage en étapes et leur organisation, tout en mettant en jeu des connaissances dans un champ des mathématiques scolaires ? Comment s'articulent, dans la définition de la situation et dans l'activité des élèves, ces deux niveaux, l'un relevant de l'algorithmique et l'autre d'un champ plus classique ? Le cas de l' « algorithme »[3] de Kaprekar que nous allons exposer plus loin nous a semblé intéressant à considérer dans cette problématique. D'une part, comme nous allons le voir, l'expression en langage algorithmique à partir de la donnée d'un traitement « manuel », impose l'explicitation non triviale d'un découpage en étapes et de l'organisation de ces étapes. D'autre part, cette explicitation met en jeu des connaissances en arithmétique. Notons que le traitement n'est pas une réponse à un problème. Il trouve son intérêt dans les propriétés de la suite de résultats obtenue en l'itérant. L'enjeu sous-jacent à l'expression de ce traitement dans un langage algorithmique est donc cohérent avec notre problématique de faire prendre conscience aux élèves des contraintes d'explicitation inhérentes à ce type de langage.

Nous terminons cette introduction en présentant et en étudiant cet « algorithme », puis nous développons un cadre théorique en psychologie de la programmation et relativement aux situations d'apprentissage, avant d'aborder la conception et l'étude d'une situation répondant à notre question.

1.1 Présentation de l' « algorithme »

L'« algorithme » de Kaprekar a été proposé pour les nombres de quatre chiffres, par le mathématicien qui lui a donné son nom ; il peut être généralisé à tous les nombres. Un entier p strictement positif étant choisi, nous considérons les nombres entiers naturels inférieurs à 10^p écrits avec p chiffres dans la numération de position en base 10. L'« algorithme » utilise une fonction K associant à un de ces nombres n un autre nombre $K(n)$ généré de la façon suivante. On forme le nombre n_1 en arrangeant les chiffres du nombre n dans l'ordre croissant et le nombre n_2 en les arrangeant dans l'ordre décroissant. On pose $K(n) = n_2 - n_1$. Il est facile de voir que $K(n)$ est nécessairement positif et inférieur à 10^p et donc il est possible d'itérer. On poursuit l'itération jusqu'à obtenir un nombre déjà obtenu. L'ensemble des nombres positifs et inférieurs à 10^p étant fini, le traitement termine nécessairement.

1.2 Étude selon le nombre de chiffres choisi[4]

- Un chiffre : le point fixe 0 est obtenu au premier pas.
- Deux chiffres : au premier pas le nombre obtenu est le point fixe 00 si tous les chiffres sont égaux, sinon c'est un multiple non nul de 9 et le traitement termine selon le cycle, 09, 81, 63, 27, 54, 09, …

3 Nous distinguerons le traitement informel que nous désignerons par « algorithme » de Kaprekar, de l'algorithme proprement dit, c'est-à-dire l'expression dans un langage algorithmique, telle qu'elle est demandée aux élèves.

4 Nous nous limitons à la base 10. Pour chaque domaine considéré, une preuve peut être obtenue en traitant tous les cas. Il est aussi possible de prouver algébriquement des propriétés des éléments du sous-domaine des résultats après une itération et donc de limiter le traitement à ce sous-domaine, comme nous le montrons pour les nombres à deux ou trois chiffres. Pour les nombres à quatre chiffres, une preuve sans ordinateur a été donnée par Kaprekar.

- Trois chiffres : au premier pas le nombre obtenu est le point fixe 000 si tous les chiffres sont égaux, sinon c'est un multiple de multiple de 99 inférieur à 1000 et en considérant ensuite chacun de ces multiples, on constate que le point fixe 495 est toujours atteint.
- Quatre chiffres : on montre de même que dans tous les cas, un des points fixes 0000 ou 6174 est atteint.
- Cinq chiffres : on montre que l'on atteint le points fixe 00000 ou que le traitement termine sur un des cycles {53955, 59994}, {71973, 83952, 74943, 62964} ou {75933, 63954,61974, 82962}.

2. Cadre théorique

Notre cadre doit, d'une part, fournir des outils pour aborder l'activité de débutants confrontés aux difficultés que nous avons signalées, et, d'autre part, nous permettre de concevoir et d'évaluer une situation d'apprentissage.

2.1 Schémas et plans dans la construction d'un algorithme

Nous nous référons à un texte (Rogalski et Samurçay 1990) s'inscrivant dans le champ de la « psychologie de la programmation ». Ce choix de la psychologie de la programmation comme cadre pour analyser des activités algorithmiques nécessite une explication. Pour les activités que nous visons, un texte exprimant un traitement est un algorithme s'il est le support d'une analyse de ses propriétés et donc respecte des contraintes formelles ; c'est un programme si l'on s'intéresse à son exécution sur un dispositif ad hoc, ce qui suppose qu'il respecte des contraintes liées à ce dispositif. A un certain niveau, il est possible et souhaitable de distinguer les deux points de vue, les propriétés d'un traitement s'analysant de façon formelle en fonction d'axiomes, sans référence à une exécution. Ce type d'analyse n'est pas accessible à des débutants. Les contraintes d'expression d'un algorithme ne se conçoivent que progressivement comme un système d'axiomes : la référence à un dispositif exécutant le traitement est essentielle ; algorithmique et programmation vont de pair.

Les travaux de Rogalski et Samurçay (1990) portent sur un public constitué d'adultes et d'étudiants novices et sur des tâches analogues à celles qui sont proposées aux élèves de lycée en algorithmique, domaine où, comme nous venons de le dire, la distinction algorithme-programme a finalement peu de substance. Les auteurs définissent les notions de schéma, c'est-à-dire de structures utilisées dans le traitement des informations pour atteindre des objectifs à petite échelle, et de plan, c'est-à-dire d'ensemble organisé de schémas s'adaptant à cette organisation. Ceci permet d'analyser des choix et gestes non triviaux dans la construction d'algorithmes.

Par exemple, considérons la tâche d'écrire un algorithme sommant les carrés des entiers inférieurs à un entier n donné[5]. Sa résolution implique un choix stratégique non trivial pour des débutants : soit (1) on calcule dans une première itération les n carrés puis on en fait la somme à l'aide d'une seconde itération, soit (2) on forme une seule itération où l'on « accumule » les carrés calculés à l'intérieur du corps de l'itération. Dans le choix (1), le plan comporte deux schémas de traitement itératif. Le problème est d'organiser ces deux schémas, en construisant une structure de donnée servant à leur interface c'est-à-dire une structure complexe (tableau ou liste) pour conserver les valeurs des carrés. La

5 Les auteures se situent implicitement dans un paradigme de programmation « impératif » (http://fr.wikipedia.org/wiki/Paradigme_(programmation)). Nous faisons le même choix. Discuter les conséquences de ce choix pourrait faire l'objet d'un article en lui-même.

structure itérative du choix (2) met en jeu un schéma général d'accumulation, qui, si le sujet en dispose, peut être spécifié pour le problème posé en coordonnant le calcul du carré du nombre courant et l'accumulation de ce carré à la somme en cours de construction.

Il en résulte que des tâches en algorithmique ou programmation nécessitent de penser à la fois le plan général et les schémas qu'il organise : on ne peut pas commencer à construire l'algorithme sans avoir fait le choix d'un plan, mais le choix d'un plan implique une idée des schémas possibles. Les experts allient démarche « ascendante » (de schémas connus vers un plan les organisant) et démarche « descendante » (à partir d'un plan, il s'agit de spécifier des schémas). La difficulté pour un débutant est de ne pas disposer d'un répertoire de schémas suffisamment complet et adaptable pour imaginer un plan adéquat. Dans l'exemple ci-dessus, un débutant ne disposant pas d'un schéma général d'accumulation se déterminera pour le plan (1) le plus proche d'un traitement manuel, mais ne sera pas capable d'établir des schémas élémentaires adaptés, faute de connaître une structuration des données adéquate.

2.2 Éléments de la théorie des situations didactiques

Notre problématique énoncée ci-dessus porte sur les situations à même de conduire au choix d'un plan et la construction des schémas élémentaires (c'est-à-dire de traitements correspondant aux étapes du plan) mettant en jeu des mathématiques scolaires, et sur leur contribution aux apprentissages de ces mathématiques par les élèves.

Mettre en place et étudier une situation d'apprentissage suppose un cadre théorique adapté Nous avons souligné plus haut avec Crahay (1987) l'approche naïve du constructivisme souvent rencontrée dans les activités scolaires de programmation. La théorie des situations didactiques (TSD) a prouvé son efficacité comme approche rigoureuse des situations d'apprentissage et l'ingénierie didactique s'est imposée comme une méthode pour construire et tester des situations à l'aide d'observations contrôlées. Nous ne prétendons pas ici atteindre un niveau optimum de rigueur pour deux raisons : le travail sur lequel se base cet article est une première tentative d'ingénierie dans un domaine encore peu exploré[6] ; d'autre part, notre travail se situe en fin de lycée, niveau proche de l'Université, qui présente des spécificités notées par Artigue (2014 p. 136) particulièrement le fait que le milieu comporte de nombreux éléments non matériels (signes, résultats mathématiques…) et le rôle essentiel que doit jouer l'enseignant pour rendre ce milieu productif.

Parmi les concepts les plus importants de la TSD, nous retenons tout d'abord la confrontation à un milieu antagoniste, face auquel l'élève doit adapter ses connaissances (Brousseau 1987). Dans le cas d'une tâche de construction d'un algorithme, le milieu est constitué du langage algorithmique et des objets à traiter. Le langage exerce des rétroactions résultant de contraintes d'expression, définies de façon formelle, ou en référence à un dispositif sur lequel exécuter l'algorithme. Par ailleurs, comme la situation que nous visons doit mettre en jeu des mathématiques scolaires, les objets à traiter sont des objets mathématiques et donc les rétroactions renvoient à leurs propriétés

6 Samurcay et Rouchier (1990) et Nguyen (2005) ont exploré ce domaine dans d'autres contextes curriculaires. Plus récemment Modeste (2012) se situe dans le même contexte curriculaire que nous, mais n'aborde pas les questions que nous traitons dans cette article. Belpaire (2013) rencontre ces questions à travers une proposition de « reprise de l'algèbre par l'introduction de l'algorithmique ».

mathématiques. Relativement à la planification, plans et schémas[7] doivent émerger d'une confrontation aux contraintes du langage. Les questions posées aux élèves sont aussi des éléments critiques du milieu pour installer de façon productive cette confrontation.

Le contrat didactique est une autre notion importante. Une tâche relative à l'expression d'un traitement dans un langage algorithmique impose de travailler de façon non linéaire, d'une part à la construction de schémas, et d'autre part à l'élaboration d'un plan. Une confrontation a-didactique aux contraintes d'expression dans un langage algorithmique, peut amener en premier lieu les élèves à identifier des éléments non explicites dans un traitement « manuel ». Nous attendons ensuite, d'une discussion menée par l'enseignant, que cette confrontation fasse émerger une ébauche du plan, et des schémas compatibles avec ces éléments : il s'agit d'éviter que les nécessaires allers et retours entre schémas et plans ne s'éternisent. Ensuite, l'écriture des schémas et leur organisation dans le plan peuvent faire à nouveau l'objet d'un travail en a-didacticité. Il en résulte un découpage en phases ayant des contrats didactiques différents. Nous caractériserons ces différents contrats en spécifiant pour chaque phase ce qui est à la charge respectivement des élèves et de l'enseignant.

Nous retenons l'ingénierie didactique comme une technique puissante pour élucider des phénomènes didactiques et tester des hypothèses de recherche (Artigue 2014, p.138). Elle implique une analyse épistémologique, cognitive et didactique du savoir en jeu, la détermination de variables didactiques, des choix relatifs à ces variables, l'analyse a priori des conséquences de ces choix dans les phases a-didactiques, et une analyse a posteriori d'une mise en œuvre permettant de conclure sur les phénomènes didactiques et hypothèses de recherche en jeu.

3. Milieu et planification dans l'« algorithme » de Kaprekar

Dans cette partie, nous allons spécifier notre analyse du savoir en jeu par une étude de l'« algorithme » de Kaprekar.

Précisons le milieu : les objets sont des nombres entiers et la tâche met en jeu deux représentations, l'une comme donnée numérique sur laquelle exécuter des opérations arithmétiques, l'autre comme un n-uplet de chiffres, permettant de leur rangement (tri). Le traitement sur ces représentations suppose que le langage algorithmique, autre élément du milieu, comporte, outre les structures élémentaires (affectation, lecture, écriture, alternative, itération), des primitives de soustraction et de comparaison opérant sur des données de type numérique.

Deux plans possibles se distinguent par la façon dont la représentation comme suite de chiffres intervient dans le traitement :

- Dans l'un des plans, le nombre donné et son image par *K*, ainsi que les nombres intermédiaires sont traités seulement comme des n-uplet de chiffres. Les chiffres peuvent être triés tels quels, mais il faut concevoir un schéma pour effectuer la soustraction de deux nombres sous la forme d'un n-uplet de chiffres.
- Dans l'autre plan, le traitement prend un nombre en entrée et en extrait le n-uplet de ses chiffres seulement pour pouvoir les ordonner ; il forme ensuite les deux nombres intermédiaires pour les soustraire. Ce plan implique de disposer de schémas de conversion d'un nombre vers le n-uplet de ses chiffres (extraction) et de la conversion inverse ainsi que d'un schéma de tri.

7 Dans la suite de l'article, « plan » et « schéma » sont utilisés au sens de Rogalski & Samurçay (1990).

La construction de l'algorithme nécessite donc de choisir un plan. Ce choix n'existe pas dans un traitement « manuel », où les deux représentations d'un nombre ne sont pas distinguées et donc la construction de l'algorithme suppose une prise de conscience de ces deux représentations et des traitements que chacune permet : la représentation sous forme de nombre ne permet pas le tri des chiffres, et la soustraction implémentée dans le langage n'opère pas sur des n-uplets de chiffres.

Des connaissances en numération décimale de position sous-tendues par des connaissances en arithmétique sont nécessaires pour concevoir les différents schémas composant chacun des deux plans : un schéma de soustraction opérant sur un n-uplet de chiffres dans le premier plan et des schémas de conversions dans le second plan. La construction de l'algorithme et les choix devant lesquels elle place l'élève sont donc cohérents avec la question que nous souhaitons étudier.

4. Étude de ressources

Afin de dégager des variables concernant le découpage en phases d'une situation d'apprentissage et le contrat didactique dans chacune des phases nous allons étudier quelques ressources exploitant l'« algorithme » de Kaprekar choisies sans souci d'exhaustivité. Nous avons trouvé le problème de l'écriture de l'« algorithme » de Kaprekar dans le contexte du curriculum du lycée, dans un manuel (Chareyre et al 2012) désigné par Math'X dans la suite, et dans une ressource web produite par un groupe de l'IREM Franche-Comté et publiée sur le site de cet IREM (IREM FC 2010). Dans les deux cas, l'« algorithme » est spécifié pour des nombres à trois chiffres, cas, où, nous l'avons vu, le traitement termine sur un point fixe (000 ou 495).

4.1 Étude du manuel Math'x

Les tâches proposées

Dans une rubrique « résolution de problème », le manuel propose une série de tâches sous le titre « La recette de Kaprekar ». Le titre est suivi de l'énoncé d'un objectif : « travailler l'écriture dans le système décimal et sur l'extraction des chiffres d'un nombre de façon algorithmique ». La « recette » est donnée dans un encadré. Le texte précise d'emblée un découpage en une initialisation (« Choisir un nombre à trois chiffres ») et une itération comprenant 4 étapes correspondant au second plan ci-dessus : (1) « extraction des chiffres » (2) « prendre les chiffres et les classer du plus grand au plus petit pour écrire un nombre M0 » (3) « reprendre les chiffres et les classer du plus petit au plus grand pour écrire un nombre m0 » (4) « calculer la différence N1=M0-m0 »…

La tâche se divise en trois sous-tâches A, B, C correspondant respectivement à l'exécution « manuelle » pour quelques valeurs, à son implémentation dans un tableur et à une démonstration[8]. La sous-tâche A se termine par une demande de conjecture « Qu'observez-vous ? ». La sous-tâche B consiste en la création guidée d'une feuille de calcul dans un tableur. Il est précisé que le chiffre des unités est le reste dans la division par 10 et la primitive correspondante du tableur est indiquée. Il est ensuite demandé de calculer « le nombre formé des deux premiers chiffres » puis de « finir l'étape 1 ».

8 La sous-tâche C concerne la démonstration de la terminaison sur un point fixe, tâche que nous ne considérons pas dans notre étude. Cette tâche pourrait s'insérer dans notre problématique générale, mais elle renvoie à une question un peu différente de la question posée dans cet article : comment l'écriture de l'algorithme peut-elle contribuer à la construction d'une preuve par les élèves ? Cette question est étudiée par Laval (à paraître) dans le cadre de son travail de thèse.

Pour la seconde et la troisième étape, il est indiqué d'utiliser la primitive GRANDE.VALEUR qui prend comme arguments une plage de cellules et un entier, et renvoie la valeur dont l'ordre est l'entier dans la plage de valeurs ordonnée de la plus grande à la plus petite. L'organisation de la feuille de calcul est aussi indiquée par une copie d'écran, avec les valeurs obtenues pour la donnée 517, la formule à entrer à l'étape 2 étant indiquée en surimpression. Seule la conversion de la suite de trois chiffres à la donnée numérique du résultat reste à la charge des élèves.

Notre analyse

Nous nous centrons sur les tâches effectivement à la charge des élèves dans l'expression du traitement dans le tableur (tâche B)[9] et qui dépassent la simple exécution technique. L'identification des étapes est donnée d'emblée par le manuel, avant même l'exécution manuelle alors que les étapes d' « extraction » des chiffres et de classement de ces chiffres ne sont pas pertinentes pour un opérateur humain. Le codage de cette étape d'extraction est partiellement à la charge des élèves. La fonction permettant d'isoler le chiffre des unités dans le tableur est donnée. En revanche implicitement le manuel suppose que les élèves reconnaîtront le nombre formé des deux premiers chiffres comme le quotient euclidien du nombre initial par 10 et connaîtront une formule tableur pour ce calcul, puis qu'ils feront le même travail sur ce quotient pour obtenir les chiffres des unités et de centaines. Le tri des chiffres dans l'ordre décroissant est donné par le manuel comme une organisation de cellules dans le tableur et l'indication de la formule à entrer dans les cellules. La conversion d'une suite de chiffres vers un nombre est à la charge des élèves. L'analyse montre donc que le manuel accorde une place importante aux étapes de conversion comme tâches à la charge des élèves. La nécessité de ces étapes n'est en revanche pas problématisée, et le tri des chiffres est traité comme une simple étape technique. D'un côté, l'accent mis sur les étapes de conversion est cohérent avec l'objectif annoncé de travailler l'écriture dans le système décimal et l'inscription de ces tâches dans un chapitre d'arithmétique consacré notamment à la division euclidienne. D'un autre côté, la situation de résolution de problème est très pauvre, le manuel prenant en charge le découpage en étapes et la réalisation de la majorité des étapes.

4.2 La ressource de l'IREM de Franche-Comté

La ressource web est organisée en cinq volets : énoncé ; intentions ; fiche élève ; fiche professeur ; expérimentation. Ce dernier volet présente trois mises en œuvre en classe.

L'énoncé

Il présente l'« algorithme » comme une itération. Le corps de cette itération s'énonce ainsi : « On considère les chiffres de D. On forme le nombre P en arrangeant ces chiffres dans l'ordre croissant et le nombre G en les arrangeant dans l'ordre décroissant. On pose K(D) = G - P. » Le découpage en étapes est donc partiel, puisque la même seconde phrase groupe les étapes de tri et de conversion inverse. Il correspond plutôt à un traitement « manuel ».

La fiche d'intentions

Cette fiche explique certains choix, notamment la limitation à trois chiffres et les objectifs qui peuvent être poursuivis aux différents niveaux du lycée, de la Seconde à la Terminale.

9 L'expression d'un traitement dans un tableur prend une forme différente de l'expression dans un langage « classique », Néanmoins, on y distingue bien une structuration en étapes, pertinente pour notre analyse,

Elle situe « l'extraction des chiffres » comme une tâche intéressante mettant en jeu des notions mathématiques (division euclidienne, numération décimale) et algorithmique (boucle pour).

La fiche élève

Cette fiche est découpée en quatre exercices. Le premier propose comme tâche le tri de deux puis trois nombres. Le second exercice porte sur la division euclidienne de deux entiers naturels. Les tâches visent à s'approprier la notion puis à écrire un algorithme donnant le quotient et le reste dans la division euclidienne de deux entiers naturels. Les troisième et quatrième exercices demandent de construire l'extraction des chiffres d'un nombre à deux puis à trois chiffres. Les objectifs ne sont pas précisés, mais on comprend qu'il s'agit de travailler séparément les étapes conçues par les auteurs comme les plus importantes.

La fiche professeur

Cette fiche est un document de 12 pages qui propose d'étudier l'« algorithme » de Kaprekar pour un nombre à trois chiffres, tout en précisant que « certains algorithmes choisis sont facilement adaptables à un entier quelconque ». Il s'agit de l'exposé très détaillé de différents algorithmes pour (1) la dissociation des chiffres composant le nombre (2) la détermination du plus petit et du plus grand entiers que l'on peut former avec une liste de chiffres, et (3) l'itération du processus. Les algorithmes sont exprimés dans un langage dont les spécificités ne sont pas précisées. La lecture de ces algorithmes montre que, outre les structures et primitives que nous avons supposées plus haut, le langage comporte une structure de liste d'entiers et des primitives donnant le produit, la somme, le maximum et le minimum d'une telle liste, ce qui, comme le document le précise à la page 2, simplifie notablement le tri dans le cas d'un nombre à trois chiffres. Globalement la fiche professeur semble retracer les différents essais réalisés par le groupe, sans beaucoup de réflexion sur l'expression et la validité des algorithmes. Tout comme l'énoncé, la structure du document ne met pas l'accent sur le tri des chiffres comme étape autonome.

Le volet « expérimentation »

Ce volet propose deux comptes rendus de travaux en Seconde et un en Première.

Le travail proposé par l'enseignant d'une des classes de Seconde

Il est constitué de cinq exercices. Les algorithmes sont exprimés dans un langage dont les spécifications ne sont pas précisées. Comme on le verra, le langage utilise la division et la fonction partie entière et n'utilise pas de listes. Le premier exercice est une première prise de contact avec l'« algorithme », à partir de l'énoncé donné dans la ressource, que les élèves doivent exécuter à la main. Cet exercice est suivi de l'établissement d'une conjecture en classe. Le second exercice présente aux élèves un algorithme de structure alternative impliquant trois variables « réelles » x, y et z, la condition étant ($x < y$) et les deux branches étant constituées de l'affectation à z respectivement du contenu de x et de y. Les élèves doivent comprendre les effets, c'est-à-dire que z se voit affecté la plus grande des deux valeurs. Dans une seconde question, les élèves sont invités à écrire un algorithme renvoyant le plus grand de trois entiers. Ils proposent ainsi plusieurs stratégies basées sur des alternatives consécutives ou imbriquées. Le troisième exercice demande d'élaborer un algorithme permettant de déterminer le chiffre des unités d'un nombre d'un entier quelconque. Cet exercice a posé de nombreuses difficultés et l'algorithme final a été

élaboré en classe avec l'aide de l'enseignant. Une première version itère la soustraction de 10, tandis qu'une version « améliorée » utilise la division et la partie entière. Le quatrième exercice fait référence aux exercices précédents en demandant comment obtenir aussi le chiffre des dizaines et le chiffre des centaines d'un nombre entier à trois chiffres puis comment classer par ordre décroissant trois nombres donnés. Ensuite il est demandé d'en déduire un algorithme qui permettrait « d'automatiser ». Le cinquième exercice conduit à une preuve algébrique de la terminaison, sans référence à l'algorithme écrit à la quatrième question. Aucune indication n'est donnée quant au travail effectif des élèves sur ces deux exercices.Globalement ce compte rendu est conforme aux intentions exprimées dans les autres volets. La décomposition en étapes n'est pas problématisée et l'accent est mis sur les schémas élémentaires, non directement identifiables dans la décomposition proposée. Les schémas élémentaires sont composés dans l'algorithme. Ainsi la démarche peut être qualifiée d'« ascendante ». Les quelques indications données sur les élèves montrent que les tâches, même dirigées, ne sont pas triviales.

Le travail proposé par une enseignante d'une autre classe de Seconde

Cette enseignante a proposé une activité en quatre temps, un premier exercice à préparer en temps libre suivi d'un premier bilan en classe, puis un second exercice donné en temps libre et enfin un dernier bilan en classe. L'énoncé a la même structure que le volet énoncé de la ressource. Il donne des titres aux trois parties, entrée, traitement, sortie, et propose de «répéter 5 fois » pour la partie traitement. Les élèves sont invités (1) à déterminer les variables informatiques utilisées dans cet algorithme (2) à préciser les valeurs de ces variables pour différentes exécutions « manuelles » puis (3) à émettre une conjecture.

L'accent mis sur les variables pourrait suggérer que le professeur a voulu engager les élèves à prendre en compte les contraintes du langage, notamment l'introduction de variables pour stocker les chiffres du nombre. Ce n'est pas le cas puisqu'il leur a été demandé simplement de repérer des identificateurs de variable dans l'algorithme proposé qui formalise un traitement où le rangement des chiffres est implicite. Le premier bilan en classe a porté sur la conjecture et non sur les variables nécessaires. Un groupe d'élèves a proposé de tester tous les cas possibles avec un tableur. Les élèves ont alors été très guidés par l'enseignante qui leur a présenté plusieurs fonctions du tableur. L'« extraction des chiffres » est mentionnée seulement pour préciser qu'elle a été effectuée par une fonction de traitement de chaîne de caractère (STXT, sous-texte). Implicitement, il s'agit d'une étape technique pour l'enseignante. Concernant la conversion inverse, l'enseignante observe que l'écriture d'un nombre à partir des trois chiffres présente une réelle difficulté. Il s'agit d'une tâche laissée ouverte comme dans le Math'*x* et l'observation montre que la mise en œuvre de la numération décimale dans un algorithme n'est pas triviale.

Ainsi, l'enseignante a pris à sa charge les deux premières étapes : passage du nombre aux chiffres et tri des chiffres, en suggérant l'utilisation de certaines fonctions du tableur. Par ailleurs, les traces de l'enseignante n'indiquent pas si l'idée de la décomposition en chiffres vient d'elle ou des élèves. Avec des formes de travail différentes, la démarche est analogue à celle de l'enseignant précédent. L'accent est mis sur les « variables », c'est à dire différentes données numériques intervenant dans le traitement, mais comme la représentation des données n'est pas discutée, l'« algorithme » est adapté à une exécution manuelle. Comme dans Math'*x*, le passage vers le tableur est vu comme une tâche technique, minorant la question des conversions et du tri. L'ensemble des tâches est en fait dirigé vers la preuve de la conjecture. L'enseignante n'indique pas si le travail sur le tableur a conduit à une prise de conscience de l'écriture décimale facilitant cette preuve.

Le travail proposé par un enseignant en Première S

Cet enseignant a choisi de travailler autour de l'« algorithme » de Kaprekar pendant trois séances. La première séance reprend les tâches des deux premiers exercices de la fiche élève de la ressource, la seconde et la troisième correspondant respectivement aux exercices 3 et 4. Le document se présente comme un compte rendu du travail des élèves, l'auteur s'en tenant implicitement aux objectifs et choix énoncés dans les autres volets. Le choix est cependant fait du logiciel AlgoBox pour l'expression des algorithmes. Nous présenterons plus loin ce logiciel qui produit un texte exécutable. Les primitives utilisées ne sont pas précisées avec ce choix. On apprend au fil du rapport que le langage comprend une fonction « division entière » et une fonction « partie entière » qui sont mises en valeur pour exposer des variantes.

La première séance est donc constituée d'une première rencontre avec l'« algorithme », et de l'écriture d'un algorithme de tri de trois chiffres. Le compte-rendu montre de nombreuses difficultés liées à la compréhension du langage soulignées dans l'introduction de cet article. Les élèves qui parviennent à écrire un algorithme distinguent les 6 cas à l'aide d'alternatives à une branche (pas de sinon). L'enseignant ne s'en satisfait pas. Il demande d'utiliser des alternatives complètes et laisse les élèves terminer l'algorithme à la maison. Mais les traces ne présentent pas les travaux effectués par les élèves, et l'enseignant précise qu'il distribue un corrigé à la séance suivante. Ainsi, bien que les élèves aient cherché à construire cet algorithme, l'algorithme final est fourni par l'enseignant.

La seconde séance porte sur la décomposition du nombre en chiffres. Cela semble aussi poser de nombreuses difficultés aux élèves. Selon l'enseignant, deux idées « émergent finalement »: tant que $x>10$, x prend la valeur $x-10$ et l'utilisation de la partie entière pour trouver le chiffre des centaines, mais le compte rendu ne dit pas clairement si cette « émergence » a été réellement à la charge des élèves. L'enseignant propose ensuite une troisième stratégie utilisant la division euclidienne.

La dernière séance à pour but de reconstituer l'algorithme à partir des travaux faits précédemment. Contrairement à ce qui a été observé dans la classe précédente, cette fois-ci la reconstitution du nombre ne pose que peu de problèmes. Nous supposons que cela est dû au fait que le passage du nombre aux chiffres a ici été traité en profondeur et d'un point de vue de la numération, alors que l'enseignante précédente avait utilisé la conversion implicite dans le tableur (fonction STXT). Les élèves obtiennent ainsi un algorithme fonctionnel.

Ainsi, comme dans les autres classes, l'identification des étapes n'est pas à la charge des élèves. Le script des exercices de la fiche élève est suivi plus scrupuleusement que par les autres enseignants et l'accent est mis davantage sur la construction de l'algorithme. On voit comment ce script s'inscrit dans une démarche « ascendante ». La programmation des étapes et leur organisation dans un algorithme semblent être laissées à la charge des élèves mais de nombreuses interventions de l'enseignant ont été nécessaires.

4.3 Synthèse

Dans le tableau 1, nous mettons en évidence les tâches laissées à la charge des élèves et celles à la charge de l'enseignant dans les différents énoncés et traces de pratiques que nous avons étudiés. Nous voyons ainsi que le manuel est celui qui laisse le moins de tâches à la charge des élèves. Dans les séances dirigées par les enseignants, le travail à la

charge des élèves se trouve également réduit dans certaines phases par les suggestions ou les corrections données par les enseignants qui limitent la réflexion personnelle de l'élève.

	Identification des étapes	Décomposition du nombre	Tri des chiffres	Reconstruction du nombre	Construction de l'algorithme
Math'x	Donnée par le manuel	Partiellement à la charge des élèves. L'idée est donnée par le manuel, l'élève n'a qu'à justifier, la fonction MOD est suggérée	Guidé par l'énoncé, la fonction GRANDE.VALEUR est indiquée	A la charge des élèves	Pas à la charge des élèves
2nde 1	À la charge de l'enseignant	Très guidée par les énoncés et les corrections	Guidé par les énoncés	À la charge des élèves	Très guidée par l'énoncé
2nde 2	Suggérée par les indications de l'enseignante	Suggérée par les indications de l'enseignante : présentation des fonctions SXT et ENT	Suggéré par les indications de l'enseignante : présentation de la fonction GRANDE.VALEUR	À la charge des élèves mais présente de nombreuses difficultés	À la charge des élèves
1ère S	À la charge de l'enseignant	À la charge des élèves, mais c'est l'enseignant qui donne l'algorithme final	À la charge des élèves, mais c'est l'enseignant qui donne l'algorithme final	À la charge des élèves	À la charge des élèves

Tableau 1. *Tâches laissées à la charge des élèves et celles à la charge de l'enseignant*

5. Ingénierie didactique

5.1 Choix généraux

Le choix central de l'ingénierie est d'amorcer un travail des élèves sur la planification à partir de l'identification des variables nécessaires au traitement. Contrairement à l'enseignante de la classe de Seconde dont le compte rendu est analysé ci-dessus, nous demanderons aux élèves que les variables permettent effectivement le traitement par un dispositif informatique. Il s'agit de faire apparaître la nécessité de variables représentant le n-uplet des chiffres du nombre courant, d'où découle la nécessité d'étapes de conversion et de rangement non distinguées dans un traitement « manuel ». Nous viserons à ce que les élèves comprennent les étapes comme de véritables modules autonomes qu'il s'agit de coordonner dans l'algorithme. Nous concevons cette démarche comme une mise en œuvre de l'articulation entre « plans et schémas » développée dans la seconde section de l'article.

L'« algorithme » de Kaprekar peut être exploité à différents niveaux, du collège au supérieur. Nous choisissons quant à nous la fin du lycée (terminale S). Ce choix est motivé par l'insertion de cette ingénierie dans une ingénierie plus large visant à un travail sur l'« algorithme » dans une démarche de preuve mettant en jeu des connaissances en arithmétique (voir note page 51). Par ailleurs, nous avons indiqué que les difficultés des élèves concernaient deux aspects : (1) la compréhension des contraintes d'un langage algorithmique (2) le découpage en étapes et leur organisation lors de la construction d'algorithmes, et que nous souhaitons nous concentrer ici sur le second aspect.

Nous choisissons par conséquent de travailler avec des élèves ayant déjà assimilé certaines contraintes d'expression d'un langage algorithmique.

Nous choisissons de demander que l'algorithme itère le traitement jusqu'à ce qu'un nombre soit à nouveau obtenu. L'insertion du traitement dans une itération impose en effet d'avoir le même type de variables en entrée et en sortie de l'itération. Elle invalide par conséquent un traitement qui prendrait un n-uplet en entrée et rendrait un nombre en sortie, évitant ainsi l'étape d'extraction des chiffres. C'est aussi l'occasion d'étudier chez des élèves ayant travaillé en algorithmique depuis le début du lycée comment des aptitudes concernant l'aspect (1), notamment relatives à l'arrêt de l'itération, interagissent avec l'aspect (2).

Nous choisissons comme langage pour exprimer l'algorithme, celui du logiciel Algobox. Algobox se présente comme un logiciel d'aide à l'élaboration et à l'exécution d'algorithmes. Les caractéristiques que son concepteur met en avant sont les suivantes :

- « Le code de l'algorithme est construit à partir d'un mini-langage algorithmique ("pseudo-code") qui se veut simple à comprendre ».
- « Le code de l'algorithme se construit pas à pas de façon hiérarchique et structurée grâce à des instructions de base que l'on insère en cliquant sur les boutons de l'interface : l'utilisateur se concentre ainsi sur l'algorithme lui-même et il est fortement incité, par le fonctionnement même du programme, à faire preuve d'un minimum de rigueur. » (Brachet 2009)[10]

Les caractéristiques importantes pour nous sont les suivantes :

- Les élèves avec lesquels nous allons travailler sont habitués à utiliser ce logiciel, ce qui va dans le sens de nous concentrer sur la planification plutôt que sur les difficultés relatives au langage.
- Les élèves ont aussi été initiés au tableur, mais les usages de ce logiciel dans les ressources étudiées montrent que l'organisation en tableau ne se prête pas à l'identification des variables, ni à une approche modulaire qui sont pour nous les éléments déterminants pour la planification.
- Le langage (ou « code » dans la présentation d'Algobox) est exécutable. Ceci permet au texte produit par les élèves d'être considéré comme un « programme » aussi bien que comme un « algorithme ». Comme nous l'avons dit plus haut, quand un texte peut être ainsi considéré sous ces deux angles, comme c'est le cas dans Algobox, les contraintes du langage sont mieux comprises comme liées à un dispositif opérant en différé et l'accent peut être mis aussi sur la lisibilité pour la réflexion et l'évaluation.
- Nous nous limitons aux fonctions arithmétiques d'Algobox (opérations, partie entière, division entière, reste, comparaison) et sur les listes (maximum et minimum). Algobox dispose de fonctions de conversion et de traitement de chaînes, mais elles sont peu commodes et non connues des élèves.
- Algobox ne permet pas l'écriture de procédures ou fonctions qui sont souvent associées à une approche modulaire. Nous pensons néanmoins possible que les élèves traitent l'algorithme en jeu de façon modulaire en coordonnant plusieurs sous-algorithmes correspondant aux traitements identifiés et conçus séparément.
- En dehors de l'itération contrôlée par un compteur (boucle POUR), la seule structure itérative proposée dans Algobox est du type "test/traitement" (Tant que… faire.. .FinTantque). Le test repose alors sur une « condition de continuation » moins facile à gérer par des débutants que la « condition d'arrêt » d'une structure du type « traitement/test » (Répeter… jusqu'à…) puisqu'il faut exprimer une condition sur un

10 Une discussion sur les termes employés et les conceptions sous-jacentes sortirait du cadre de cet article.

objet qui n'a pas encore été traité (Rogalski et Samurçay, 1990). Nous constatons cependant que les langages proposés généralement pour l'initiation privilégient cette structure « test/traitement», plus générale que la structure du type "traitement/test" qui implique que le corps de l'itération est exécuté au moins une fois. Implicitement, il est décidé de proposer une seule structure et donc choisir la plus générale.

Nous choisissons de faire travailler les élèves avec des nombres à trois chiffres. Ce choix est motivé par plusieurs raisons. Tout d'abord, nous évitons de traiter le cas des nombres à deux chiffres, d'une part parce que le nombre d'entiers à tester est faible et qu'un traitement de tous les cas peut être effectué à la main par les élèves ; d'autre part il nous semble plus intéressant de mettre en évidence un point stationnaire plutôt qu'un cycle. En effet, les élèves sont plus familiers des points stationnaires ; lors de l'étude de la suite de Syracuse par exemple, il est apparu que les cycles auxquels conduit l'exécution perturbent certains élèves qui ont du mal à admettre qu'une fois que l'on est entré dans le cycle, il se répète indéfiniment.

Nous souhaitons par ailleurs laisser à la charge des élèves l'expression de la condition de continuation de l'itération, ce qui est plus commode dans le cas d'un ou plusieurs points fixes. En effet, dans ce cas, il suffit de comparer le nombre obtenu aux points fixes, alors que pour repérer un cycle, il convient de conserver la liste des valeurs obtenues et de construire un parcours de cette liste[11]. Ceci laisse le choix entre des nombres à trois ou quatre chiffres. Nous optons pour le premier choix par souci de simplicité, mais l'autre choix ne dénaturerait pas l'ingénierie.

Dans la troisième section de cet article, nous avons distingué deux plans possibles. Dans l'un, tous les nombres intervenant dans l'algorithme sont traités seulement comme des triplets de chiffres. Dans l'autre plan, les chiffres interviennent seulement dans l'étape où ils sont ordonnés ; il met en jeu des schémas de conversion et de tri. Le premier plan évite la conversion du nombre aux chiffres et inversement, mais en soulève d'autres. Tout d'abord, cet algorithme comportant une itération, il convient d'avoir le même type de variables en entrée et en sortie de l'itération. Ainsi, si on choisit d'entrer un triplet, il faut renvoyer un triplet et non pas un nombre. Il est donc cohérent d'effectuer tout le calcul sur le triplet sans jamais reconstituer le nombre. Cela nécessite, comme nous l'avons souligné, de construire une soustraction opérant sur des triplets. Une telle construction semblant irréaliste en a-didacticité dans un temps raisonnable, à la différence des schémas de conversion et de tri, nous choisissons de favoriser le second plan tout en laissant la possibilité que le premier intervienne dans la discussion avec les élèves. C'est pourquoi l'énoncé demande que l'algorithme commence par la lecture d'un nombre à trois chiffres.

5.2 Scénario des séances en classe et analyse a priori

Voici le scénario tel qu'il a été prévu a priori. Il se déroule sur deux séances en classe entière (24 élèves) de Terminales S, spécialité Mathématiques, en salle informatique. Chaque séance dure deux heures, mais dans la seconde, seule la première heure est consacrée à l'expérimentation décrite ici, la seconde heure étant consacrée à l'ingénierie sur la preuve que nous ne considérons pas ici.

11 Prévoyant qu'une condition d'arrêt portant sur l'appartenance du résultat courant à la liste des résultats antérieurs présenterait des difficultés notables pour les élèves, nous envisageons dans l'ingénierie une condition d'arrêt reposant sur la comparaison à un ou plusieurs points fixes repérés lors d'une exploration. Comme cela ne garantit pas que l'algorithme termine, nous remettons cette question à l'étape de preuve prévue dans l'ingénierie de Laval (à paraître). Voir note page 51.

Travail préparatoire à la première séance

Avant la première séance, les élèves ont un premier travail à préparer à la maison afin de découvrir l'« algorithme » à partir d'un texte adapté à un traitement « manuel ». Ils leur est demandé d'exécuter à la main le traitement sur quelques valeurs, de suggérer un test d'arrêt pour l'itération, et, conformément à notre choix central énoncé ci-dessus, de répertorier les variables nécessaires à l'écriture de l'algorithme correspondant.

Scénario de la première séance

Lors de la première séance, les élèves sont installés par groupes de trois. Les périodes de mise en commun en classe entière et de travail de groupe alternent durant cette séance. Tout d'abord, il s'agit de faire une synthèse du travail à la maison. Les élèves discutent en groupe des réponses qu'ils ont trouvées dans ce travail (5-10 min). Un représentant de chaque groupe présente alors les résultats à l'oral et la synthèse est faite au tableau par l'enseignant (10-15 min). L'établissement de réponses aux questions « tester à la main cet algorithme sur quelques exemples, émettre une conjecture sur le résultat de cet algorithme et suggérer un test d'arrêt » est laissé à la charge des élèves ; l'enseignant se contente de mettre en commun ces réponses et de les soumettre à la validation de la classe. On s'attend à ce que la conjecture conduise à une condition d'arrêt lorsque la valeur obtenue est 495 ou zéro. En ce qui concerne la question « identification des variables à utiliser dans l'algorithme », les différentes réponses peuvent être écrites au tableau, mais l'enseignant ne cherche pas à les valider ou à les infirmer. Notre hypothèse est que la nécessité de variables pour représenter le n-uplet des chiffres découle des contraintes du dispositif vues à travers le langage Algobox.

Comme nous l'avons expliqué dans la présentation de la situation, nous souhaitons qu'un plan soit arrêté avec les élèves avant de les confronter à la conception des schémas élémentaires. L'enseignant suggère donc de segmenter l'algorithme en traitements élémentaires, mais laisse aux élèves l'identification de ces traitements. Notre hypothèse est que le repérage des deux représentations du nombre est suffisante pour que les élèves adhèrent à la suggestion de segmenter l'algorithme et identifient eux-mêmes les segments comme des traitements autonomes. Une médiation est envisagée pour que le milieu renvoie si besoin de plus importantes rétroactions aux élèves. L'enseignant peut suggérer d'appliquer l'algorithme à un nombre à 10 chiffres. Le passage par exemple de 2594131654 à 1123445569 force les élèves à expliciter leur méthode et à identifier les traitements mis en œuvre. On s'attend à ce que les traitements élémentaires suivants, non présents dans l'énoncé, soient proposés : la conversion du nombre en la suite de ses chiffres, le tri des trois chiffres, la conversion inverse. Ces trois traitements sont alors répertoriés par l'enseignant et écrits au tableau, avec, sur un exemple, les données en entrée et en sortie pour chacun de ces traitements. Cette phase est prévue pour durer 20 minutes.

Dans une seconde phase, les élèves travaillent en groupes de trois à l'écriture de sous-algorithmes correspondant aux traitements élémentaires et à leur transcription dans Algobox. Chaque groupe se consacre à un sous-algorithme : 3 groupes traitent la conversion du nombre en chiffres, 3 groupes traitent le tri des chiffres et 2 groupes traitent la reconstruction du nombre à partir de ses chiffres (conversion inverse). Nous choisissons ce dispositif car il permet de souligner à la fois l'autonomie de chaque sous-algorithme et la nécessité de les articuler, processus important pour une conception « modulaire » des algorithmes. Nous nous attendons à ce qu'il y ait plusieurs propositions

en particulier pour les deux premiers sous-algorithmes, et donc une prise de conscience de ce que des sous-algorithmes peuvent être pensés comme des « modules », par les fonctions qu'ils assurent plutôt que la façon dont ils sont écrits. L'enseignant n'intervient pas sur l'écriture de l'algorithme sur papier mais peut apporter une aide technique lors de la transcription dans Algobox. Si un groupe a terminé l'algorithme correspondant à l'étape dont il était chargé, il traite une autre étape, cela jusqu'à ce que tous les groupes aient un algorithme (dans une limite de 45 min de travail). Les travaux effectués par groupe sont alors mis en commun et validés par la classe (20 min).

Travail préparatoire à la seconde séance

Pour la séance suivante, les élèves doivent assembler ces trois sous-algorithmes afin de construire une première version de l'algorithme de Kaprekar. Il ne s'agit pas d'une simple concaténation des sous-algorithmes puisque chacun d'eux a sa déclaration de variable spécifique, et des instructions de lecture des données et d'affichage des résultats. Les élèves doivent harmoniser les identificateurs de variable et donc identifier précisément leur fonction dans les sous-algorithmes et dans l'algorithme global. Par exemple, le triplet des chiffres peut être représenté soit comme trois variables séparées, soit comme une liste. Dans le sous-algorithme de conversion, il est affiché comme résultat. Dans le sous-algorithme de classement, il est à la fois lu comme donnée et affiché comme résultat. Dans le sous-algorithme de conversion inverse, il est lu comme donnée.

Scénario de la seconde séance

Une première version de l'algorithme est mise en place collectivement à partir des travaux effectués à la maison par les élèves. On s'attend à des difficultés d'articulation entre les sous-algorithmes et dans l'écriture de l'itération. Il est prévu que des variantes soient discutées à partir des différentes réalisations des sous-algorithmes. La suite de cette séance d'une heure consiste en premier lieu pour les élèves à faire exécuter l'algorithme. Certains problèmes sont attendus lors de l'exécution de l'algorithme, par exemple si un seul des points fixes a été envisagé. Il est demandé aussi d'introduire des instructions de sorties de façon à obtenir des observables sur le comportement de la suite des résultats. Nous attendons de ce travail une compréhension plus complète de l'algorithme construit.

D'un point de vue méthodologique, les séances sont dirigées par le second auteur de cet article (désignée comme l'enseignante dans la suite), dans le cadre de son travail de mémoire (Guy, 2013)[12] et observées par le professeur de la classe. Des enregistrements sonores de la classe et de trois groupes d'élèves sont effectués, et les travaux papier des élèves et leurs algorithmes Algobox sont conservés.

5.3 Déroulement

Déroulement de la première séance

L'installation de la salle et la présentation de la séance aux élèves a duré une dizaine de minutes. Les élèves ont alors été directement répartis en 7 groupes de 3. Ils ont eu 5 minutes pour discuter du travail qu'ils avaient préparé à la maison. À l'issue de ce travail en groupe, deux représentants de groupes ont exposé leurs résultats, qui ont ensuite été discutés et complétés par les autres groupes. Le fait qu'on obtienne toujours 495 ou 0 a été rapidement conjecturé.

12 Le lecteur pourra se référer à ce mémoire notamment pour le script des échanges dans les phases collectives, des traces de ces travaux et aussi pour les documents associés aux ressources étudiées. L'énoncé des tâches données aux élèves est repris dans l'annexe de cet article.

Deux conditions d'arrêt ont été proposées : lorsque la différence de 2 nombres obtenus consécutivement est 0 et quand on obtient 0 ou 495. Les variables proposées ont d'abord été un entier A pour stocker le nombre de départ , un entier B pour stocker le nombre obtenu en arrangeant les chiffres dans l'ordre croissant, et un entier C pour stocker le nombre obtenu en arrangeant les chiffres dans l'ordre décroissant. Un élève est alors intervenu pour proposer des variables c, d, u pour stocker les chiffres des centaines, dizaines et unités de A.

> *E[13] : Ben si A c'est une variable où c'est un nombre, pour qu'il puisse le ranger dans l'ordre croissant et décroissant, faut d'abord séparer les chiffres des dizaines, des unités et des centaines. Donc il faut trois variables, un pour le chiffre des unités, un pour le chiffre des dizaines et un pour le chiffre des centaines.*

Un troisième élève s'est alors demandé si l'on pouvait faire lire directement en entrée ces trois chiffres plutôt que le nombre A. Un de ses camarades lui a rétorqué qu'il faudrait alors programmer la soustraction, avant qu'un dernier s'oppose à cela en affirmant qu'il suffirait de construire les nombres B et C à partir de ces chiffres. L'enseignante a alors fait remarquer que l'énoncé précisait « Choisir un nombre à trois chiffres » impliquant la lecture d'une seule variable et non de trois.

La planification de l'algorithme a été l'objet de la suite de cette phase collective. Comme nous l'avons expliqué dans la présentation de la situation, nous avons voulu arrêter avec les élèves un plan avant de les confronter à une spécification des schémas élémentaires. L'enseignante suggère donc de segmenter l'algorithme, mais laisse aux élèves la conception des traitements élémentaires.

> *P : Donc, là, la suite, le but de ce qu'on va faire à la suite, c'est de programmer cet algorithme d'accord. Et on va essayer après de programmer sur AlgoBox. Donc, pour pouvoir le programmer, euh, je vais, je vais vous demander, d'après vous quelles sont, quelles vont être les étapes et les petits morceaux de programmes qu'on va avoir besoin de savoir faire, de faire faire à l'ordinateur pour pouvoir programmer cet algorithme.*
>
> *E : La décomposition du nombre initial choisi en chiffre des centaines, chiffre des dizaines, chiffre des unités.*

Ainsi, le passage à un nombre à 10 chiffres envisagé comme médiation dans l'analyse a priori n'a pas été nécessaire. En fait, la recherche des variables nécessaires à la programmation de l'algorithme a constitué un travail préalable précieux. En effet, dès ce moment-là, les élèves ont perçu la nécessité d'introduire des variables pour stocker les chiffres. Ainsi, bien que l'étape d'extraction des chiffres soit implicite lors de l'exécution manuelle, la réflexion sur le choix des variables à utiliser dans un algorithme écrit pour un dispositif a permis aux élèves de la mettre en évidence. Ainsi, un élève affirme « *L'ordinateur pour arranger les chiffres en ordre décroissant, ça va être important »,* il se réfère à l'exécution par un dispositif.

L'identification du traitement suivant est également venue des élèves :

> *P : Donc étape 1 (écrit au tableau) décomposer le nombre choisi en chiffres. D'accord ? Donc qu'est-ce qu'on va avoir comme étape après ? Une fois qu'on a les chiffres, qu'est-ce qu'on va faire de ces chiffres ? Oui ?*
>
> *E : Il va falloir faire des tests les uns par rapport aux autres, pour voir lequel est le plus petit, lequel est le plus grand. Les comparer...*
>
> *P : Donc des tests par rapport aux autres pour savoir lequel est le plus petit, lequel est le plus grand. De manière générale, qu'est-ce qu'on va vouloir faire sur ces chiffres ?Oui ?*

13 Dans les extraits ci-dessous (P) désigne l'enseignante et (E) divers élèves.

E : Les classer.

P : Oui donc ?

E : Les classer dans l'ordre.

Enfin, l'identification de la reconstitution (conversion inverse) est plus laborieuse :

P : Et qu'est-ce qu'il va nous rester à faire après ?

E : La différence.

P : La différence de quoi ?

E : Reconstituer des nombres.

P : Alors, avant de faire la différence, effectivement, il va falloir reconstituer les nombres ?

E : Euh, oui, pour la différence, des nombres triés en ordre décroissant par celui trié en ordre croissant.

En effet, lors de l'exécution manuelle, le problème de la reconstitution ne se pose pas. Cependant, les élèves finissent par mettre en évidence la reconstitution en énonçant directement l'algorithme à appliquer :

E : Ben on fait fois 100, fois 10…

Ce travail en classe entière a duré 25 minutes et mis clairement en lumière la nécessité de deux représentations successives de la donnée. Il a aussi mis en évidence la nécessité de traitements intermédiaires mettant en jeu les variables c, d et u, non explicites dans le traitement « manuel ». Dans un second temps, ces différents traitements ont été explicités par les élèves et notés au tableau avec un exemple pour chacun.

Conformément au scénario il a alors été demandé aux élèves par groupe d'écrire et de tester un algorithme autonome correspondant à chaque traitement, c'est-à-dire déconnecté des autres traitements. Une élève a alors posé la question de savoir si le premier algorithme (passage du nombre aux chiffres) devait renvoyer les chiffres dans l'ordre ; l'enseignante a répondu que non. Cette partie a duré 15 minutes. Les élèves ont alors travaillé par groupes de trois à l'écriture de ces algorithmes. De nombreux groupes ont eu le temps de traiter plusieurs étapes (parfois même les 3). Seul un groupe n'a pas réussi à programmer un algorithme qui fonctionne ; s'agissant de l'étape de tri, les élèves ont persisté à vouloir écrire les conditions dans les alternatives sous la forme « $a < b < c$ », plutôt que sous la forme « ($a < b$) ET ($b < c$) ». De manière générale, la plupart des questions des élèves portaient sur ce qu'Algobox était capable de faire ou non (gérer des listes, trouver un maximum, calculer un modulo…) Cette partie a duré 45 minutes.

Enfin, les sous-algorithmes proposés ont été mis en commun en classe entière. Pour la première étape, trois types d'algorithmes ont été obtenus, un premier utilisant la fonction partie entière, un second utilisant la fonction modulo et un troisième utilisant ces deux fonctions. Pour la seconde étape, trois types d'algorithmes ont été obtenus, un premier faisant six tests successifs pour différencier les six cas (c'est celui dont la programmation n'a pas abouti), un second effectuant des tests imbriqués et stockant les résultats dans une liste, un troisième utilisant la structure de liste et les fonctions minimum et maximum d'AlgoBox. Pour la dernière étape, on a obtenu un seul type d'algorithme du type $100\,c + 10\,d + u$. Cette partie a duré 15 minutes.

Déroulement de la seconde séance

La seconde séance s'est déroulée dans des conditions un peu particulières. En effet, les élèves fêtaient Carnaval ce jour-là et étaient donc un peu moins concentrés que d'habitude. Par ailleurs, contrairement à ce qui était prévu, cette séance s'est déroulée un

mois après la première (conseil de classe et vacances empêchant deux séances consécutives). La mise en commun par groupe du travail effectué à la maison (algorithme sur papier) a duré environ 10 minutes. L'enseignante a ensuite mis en commun au tableau les travaux des élèves. Le choix des algorithmes pour chaque étape a été laissé à l'initiative de chaque groupe d'élèves. Un problème s'est posé lors de l'écriture de la condition de la boucle TantQue : il y a eu une discussion entre les élèves pour savoir si la condition de continuation était (A≠ 495 *et* A≠0) ou (A≠495 *ou* A≠0). Après une période assez confuse, l'enseignante a décidé de laisser cette question ouverte pour l'exécution dans AlgoBox, comptant sur les rétroactions de cette exécution. Cette partie a duré une quinzaine de minutes. Les élèves ont ensuite été invités à entrer cet algorithme sur AlgoBox. Malgré les difficultés rencontrées, cinq des sept groupes ont réussi à obtenir un algorithme fonctionnel après 20 minutes. Après une rapide mise au point en classe entière, où les élèves ont notamment tranché la question de la condition de continuation grâce aux rétroactions fournies par l'exécution, les élèves sont passés aux adaptations envisagées dans le scénario. Cette phase a duré environ 40 minutes. Les programmes écrits par les élèves ont ensuite été projetés au tableau devant la classe entière.

5.4 Analyse a posteriori

Analyse de la première séance

Une identification des étapes médiée par l'enseignante mais effectuée par les élèves

Le découpage en traitements élémentaires a émané des élèves, après que les variables nécessaires à un traitement algorithmique aient été identifiées. En dehors des médiations nécessaires dans une phase collective, le rôle de l'enseignante s'est limité à suggérer une segmentation de l'algorithme.

L'écriture des sous-algorithmes entièrement à la charge des élèves

L'enseignante n'est intervenue que sur des questions ponctuelles concernant le fonctionnement d'AlgoBox. Cette partie de la séance a permis aux élèves de concevoir des schémas pour chacune des étapes mises en évidence, afin de pouvoir les réinvestir lors de la conception de l'algorithme. Cependant, tous les groupes n'ayant pas avancé à la même vitesse, ils ne se sont pas tous familiarisés de la même façon avec ces schémas. La mise en commun au tableau des trois étapes a eu pour objectif de permettre à tous les élèves d'intégrer un schéma pour chaque étape.

Des questions sur le fonctionnement du dispositif

Durant la programmation des étapes par groupe de trois, l'essentiel des questions posées par les élèves concernaient le fonctionnement d'AlgoBox, et en particulier, savoir si telle ou telle fonction existait déjà ou non dans ce logiciel. Au delà de préoccupations purement techniques, ces questions révèlent également la nécessité d'avoir déjà intégré des schémas plus petits que ceux correspondant aux étapes. En effet, pour programmer la décomposition du nombre en chiffres, des micro-schémas du type "extraire une partie entière" ou "calculer le reste dans une division euclidienne" doivent pouvoir être utilisés par les élèves. Ainsi, les élèves se posent la question de savoir si AlgoBox peut effectuer ces traitements grâce à des fonctions préprogrammées, ou s'ils vont devoir programmer ces traitements eux-mêmes. De même, le groupe ayant programmé le tri des chiffres à l'aide des fonctions Min et Max a intégré pour cela des schémas faisant appel à des fonctions correspondantes d'AlgoBox, mais ne s'est pas approprié de schéma permettant de calculer la médiane d'une liste, pour laquelle une fonction existe dans AlgoBox, et qui

leur aurait permis d'identifier "le chiffre du milieu". À la place, ils ont programmé l'identification de ce troisième chiffre, mettant en œuvre un schéma d'identification d'un nombre différent de deux autres.

Algorithmique et programmation

Nous avons fait le choix de considérer les écritures dans Algobox aussi bien comme des algorithmes que comme des programmes, en le justifiant par le fait que les contraintes du langage sont mieux comprises par des débutants quant on s'intéresse aussi au texte de l'algorithme comme programme à exécuter par un dispositif. Le déroulement de la phase collective montre que ce choix y est généralement productif. Enseignante et élèves parlent de « programmer l'algorithme ». La référence au dispositif joue un rôle important notamment pour identifier la nécessité d'utiliser des variables pour les chiffres, comme en témoignent des extraits comme « *pour qu'il puisse le ranger dans l'ordre croissant et décroissant, faut d'abord séparer…* » et « *L'ordinateur pour arranger les chiffres en ordre décroissant, ça va être important* ». Néanmoins, il existe un contre-exemple : les élèves observés plus haut qui ont persisté à utiliser la forme « $a < b < c$ » comme condition dans des alternatives.

Nous analysons ceci de la façon suivante. La forme « $a < b < c$ » est courante en mathématique. D'un point de vue algorithmique, elle supposerait, d'une part, l'associativité de l'opération « < », sinon il y aurait ambiguïté entre « $a < (b < c)$ » et « $(a < b) < c$) » et, d'autre part, elle n'aurait de sens que sur des valeurs booléennes, puisque dans l'une et l'autre expression, un des opérandes est booléen. En adoptant la définition « $a < b$ est Vrai, si et seulement a est Faux et b est Vrai », le cas a=b=c=Vrai est un contre-exemple pour l'associativité. Notons que des expressions syntaxiquement correctes faisant intervenir le connecteur « < » peuvent avoir un sens. Par exemple une écriture telle que $(a < b) < (b < c)$ est syntaxiquement correcte et sémantiquement équivalente à $((a \geq b)$ et $(b < c))$. Il existe donc des arguments de type algorithmique pour rejeter la forme non parenthésée « $a < b < c$ » mais qui ne sont pas accessibles à l'élève faute d'une initiation au type booléen. L'élève ne peut comprendre ce rejet que comme une contrainte interne à l'exécution, donc de l'ordre de la programmation, alors même qu'il n'est pas gênant d'adopter cette forme dans l'écriture de l'algorithme puisque, dans le contexte, elle n'affecte pas la lisibilité de l'algorithme et même elle est plus légère puisque b n'est pas répété.

Une difficulté repérée dans la compréhension de l'itération

Notre seconde hypothèse suggérait que les élèves ne rencontraient pas de difficultés liées au langage et en particulier avec la structure itérative. Il semble cependant que la nécessité d'avoir une donnée de même nature en entrée et en sortie de la boucle ne soit pas claire pour les élèves. Considérons l'échange suivant qui est advenu après qu'un élève a proposé d'effectuer le traitement itératif avec en entrée les chiffres plutôt que le nombre (plan 1 ci-dessus) :

> *E : Dans ce cas ce serait peut-être plus utile de voir dans l'entrée les chiffres, lui dire, on rentre le chiffre des centaines, on rentre le chiffre des dizaines, on rentre le chiffre des unités. Ça empêcherait de faire une décomposition à la machine.*

Un autre élève identifie alors rapidement le problème que ce choix entraînerait:

> *E : Mais après pour le traitement de la soustraction, ça va être difficile.*

Un troisième propose alors de reconstituer le nombre :

> *E : On peut choisir le chiffre c fois 100 plus d fois 10 plus u.*

Ainsi les élèves conçoivent un traitement prenant en entrée un triplet et rendant un nombre en sortie, sans avoir conscience de ce qu'il est incompatible avec une inclusion dans une itération. De façon que cette difficulté, non centrale dans l'ingénierie, n'obère pas le déroulement de l'expérimentation, l'enseignante a rappelé que l'énoncé demandait un nombre à trois chiffres en entrée.

Analyse de la seconde séance

L'assemblage des sous-algorithmes entièrement à la charge des élèves

L'écriture de l'algorithme a été faite à la maison par les élèves, et était donc entièrement à leur charge. Comme nous l'avons dit dans l'exposé du scénario, il ne s'agissait pas d'une simple concaténation des sous-algorithmes puisque chacun de ces sous-algorithmes avait sa déclaration de variable spécifique, et des instructions de lecture des données et d'affichage des résultats. De manière générale, les élèves n'ont généralement pas rencontré de difficultés pour relier les étapes qui avaient été établies lors de la séance précédente. Les groupes qui n'avaient pas traité toutes les étapes, et en particulier ceux qui n'avaient pas écrit de sous-algorithme de classement, ont rencontré plus de difficultés que les autres. Nous avons supposé dans l'analyse a priori que l'assemblage des sous-algorithmes nécessitait l'identification des variables traitées et de leur fonction, mais pas, en principe, la compréhension du fonctionnement interne de ces sous-algorithmes. Cette hypothèse a ses limites, comme le montrent ces difficultés à assembler un sous-algorithme que l'on n'a pas soi-même travaillé.

Des difficultés liées au langage

De nombreuses difficultés se sont présentées à la classe lors de l'expression de la condition de continuation de la boucle. En effet, s'il était clair que la condition d'arrêt adéquate était (A= 0 *ou* A= 495), le passage à la condition de continuation (en fait la négation de la condition d'arrêt) n'a pas été trivial pour les élèves. Bien que nous ayons anticipé dans l'exposé des contraintes liées à AlgoBox certaines difficultés liées à la structure TantQue, ce passage a posé problème à l'enseignante et elle a dû s'en remettre à l'exécution pour que le *et* s'impose.

Synthèse

Le Tableau 2 reprend les 4 entrées du Tableau 1 qui nous ont permis de caractériser la responsabilité des élèves dans les différentes phases de la construction de l'algorithme dans les ressources étudiées. Cette responsabilité est beaucoup plus complète dans l'ingénierie. Le succès à l'étape de reconstruction du nombre est manifestement préparé par le travail sur la décomposition : les élèves ont pris conscience de la numération décimale et donc la situation implique bien des connaissances mathématiques, acquises ou réactivées dans la situation[14]. La référence au langage algorithmique c'est-à-dire à ses possibilités et contraintes, joue son rôle comme élément du milieu. Concernant les conditions, les rétroactions viennent de l'exécution (erreur de syntaxe ou boucle sans fin) plutôt que la logique du langage et ne sont pas comprises par les élèves. Ceci n'est pas central dans notre questionnement, mais montre que des ingénieries spécifiques sur les conditions seraient nécessaires.

14 Ceci est corroboré par l'ingénierie sur la preuve (voir note page 45). Laval (à paraître) analyse comment ces connaissances sont mises en œuvre pour la preuve de la terminaison sur un des deux points fixes repérés.

Identification des étapes	Décomposition du nombre	Tri des chiffres	Reconstruction du nombre	Construction de l'algorithme
À la charge des élèves (Discussion collective)	À la charge des élèves (travail par groupe)			
Préparée à la maison par l'identification des variables nécessaires à un traitement dans le langage et la suggestion de l'enseignante de segmenter l'algorithme.	Questions sur le fonctionnement d'Algobox. Plusieurs algorithmes proposés. Difficulté relative aux conditions.		Algorithme du type 100c+10d+u	Succès partiel. Difficulté à assembler un module que l'on n'a pas soi-même travaillé. Difficulté avec la négation d'une condition.

Tableau 2. Synthèse de l'analyse a posteriori

6. Conclusion

Dans l'introduction, nous avons indiqué que notre problématique portait sur la possibilité de concevoir une situation d'apprentissage imposant un découpage en étapes et leur organisation, tout en mettant en jeu des connaissances ou compétences dans un champ des mathématiques scolaires, et sur l'articulation de ces deux niveaux. Nous avons inscrit cette question dans un cadre « planification » et dans une approche TSD visant à concevoir une situation permettant la prise en charge par les élèves des différentes tâches afférentes à la planification.

La planification

Nous avons choisi de construire une situation à partir de l'« algorithme » de Kaprekar parce que le passage du traitement « manuel » sous lequel il est présenté à une écriture dans un langage algorithmique impose la conception d'un plan et de schémas qui mettent en jeu des connaissances sur la numération. Nous avons spécifié le milieu comme mettant en jeu de façon liée les rétroactions du langage algorithmique et des connaissances mathématiques. L'analyse didactique menée à partir de l'étude d'un panel de ressources existantes confirme cette mise en jeu, mais montre aussi que, faute d'une problématisation du plan et des schémas, le travail à la charge des élèves est réduit soit dès la conception de la situation, soit au cours de son déroulement (tableau 1).

Le scénario du manuel Math'*x* ne distingue pas réellement un plan et des schémas destinés à être intégrés dans un plan d'ensemble. Le découpage en étapes n'est pas discuté et ne permet pas le repérage de schémas. L'écriture dans le tableur ne les met pas non plus en évidence.

Dans la ressource de l'IREM de Franche-Comté, les exercices proposés correspondent bien à l'élaboration de schémas élémentaires, mais cette élaboration n'est pas motivée par un travail de planification préalable qui montrerait la nécessité de ces schémas. La méthode sous-jacente est donc « ascendante ».

Il résulte de notre étude que pour donner aux élèves toute leur part dans l'élaboration des algorithmes, un approche « planification » peut reposer sur un questionnement tel que :

1. Quelles données doit traiter l'algorithme et par quelles variables doivent-elles être représentées ?
2. Quels traitements particuliers de ces variables sont à prévoir ? En quoi se distinguent-ils des étapes qui servent à définir l'algorithme pour une exécution « manuelle » ?

3. Comment écrire des sous-algorithmes correspondant à ces traitements ?
4. Comment assembler ces sous-algorithmes dans un texte d'algorithme répondant aux spécifications ?

Considérer ces questions de façon purement séquentielle dans une approche strictement « descendante » n'est pas possible : la réponse à une question suppose des éléments de réponse aux questions suivantes. Par exemple, nous avons noté au \$5.2 l'intervention d'un élève proposant des variables c, d, u pour stocker les chiffres. Cet élève énonce en même temps que les variables nécessaires (question 1), la nécessité de calculer leurs valeurs (question 2) et l'hypothèse qu'un tel calcul est possible (question 3) et peut s'insérer dans un texte global (question 4). Dans l'ingénierie proposée, l'accent est mis séquentiellement sur chacune des questions, sans que les autres soient ignorées. Le rôle de l'enseignante est de recentrer sur la question en cours. Le gain en termes de prise de responsabilité par les élèves est net par comparaison avec l'approche du Math'*x* qui ne distingue pas le plan et les schémas, et avec l'approche « ascendante » de l'IREM de Franche-Comté.

Les situations d'apprentissage

La situation que nous avons conçue est basée sur un découpage précis en phases de travail, à la maison, collectives et en groupe afin de distinguer clairement les responsabilités de l'élève et celles de l'enseignant. La planification est préparée par l'identification des variables nécessaires à un traitement dans le langage, qui conduit à considérer un nombre sous deux représentations, chacune adaptée à un traitement. Après la suggestion de l'enseignante, trois schémas sont conçus comme des sous-algorithmes, deux étant des conversions et le troisième étant le tri d'un triplet de chiffres. L'analyse a posteriori (tableau 2) est dans l'ensemble conforme aux prévisions. Parmi les points qui résistent, la difficulté ressentie par certains élèves à assembler un module que l'on n'a pas soi-même travaillé montre qu'une approche modulaire, où l'on considère un sous-algorithme comme une procédure, c'est-à-dire par les données en entrée et en sortie, sans s'intéresser à son implémentation, reste une étape à franchir, au-delà de la planification. La question des situations permettant ce franchissement reste ouverte[15]. Nous avons souligné qu'Algobox ne permet pas la programmation de procédures. La disponibilité d'une structure de procédure dans le langage est une variable importante, mais il faut aussi imaginer des situations rendant nécessaire l'usage et la compréhension d'une telle structure.

Les données et leur représentation constituent aussi un point important dans l'ingénierie puisque la question initiale porte sur elles, et permet d'enclencher la démarche de planification. Le problème de la représentation des données dans le langage intervient aussi, mais cette fois dans un sens non productif, avec les difficultés des élèves concernant l'écriture de conditions. Lever ces difficultés supposerait de construire des situations amenant à considérer des conditions comme de véritables données « calculables ». Notre ingénierie reste une réalisation modeste, mais elle montre l'intérêt, dans le contexte du curriculum du lycée, d'une approche fondée à la fois sur la psychologie de la programmation comme cadre cognitif et sur la théorie des situations comme cadre didactique. D'autres études pourront suivre, approfondissant les spécificités du milieu et des contrats didactiques dans des situations de conception d'algorithmes, mais aussi de la nécessaire institutionnalisation de savoirs, point que nous n'avons pas abordé dans cet article.

15 Il est possible de voir cela comme le passage à un paradigme procédural ou fonctionnel. Voir note 3.

Bibliographie

ARSAC, J. (1988) *La didactique de l'informatique: un problème ouvert ?* Actes du Colloque francophone sur la didactique de l'informatique.

ARTIGUE, M (2014) Potentialities and limitations of the Theory of Didactic Situations for addressing the teaching and learning of mathematics at university level. *Research in Mathematics Education* 16(2), 135-138.

BARON, G.L.(1989) *L'informatique comme discipline scolaire*. PUF.

BELPAIRE, N. *Étude didactique de la reprise de l'algèbre par l'introduction de l'algorithmique au niveau de la classe de seconde du lycée français*. Thèse Université Montpellier II - Sciences et Techniques du Languedoc, 2013.

FBRACHET, P. (2009) *AlgoBox, L'apprentissage de l'algorithmique à la portée de tous.* http://www.xm1math.net/algobox/doc.html.

BROUSSEAU, G. (1988) Théorie des situations didactiques, La Pensée Sauvage.

CRAHAY, M. (1987). Logo, un environnement propice à la pensée procédurale ? *Revue française de pédagogie*, 80(1), 37-56.

CHAREYRE, B., ALVEZ, Y., WILKE, S., GUILLEMET, D., LE YAOUANQ, M-L. (2012) *Math'x - Terminale S Spécialité.* Editions Didier, Paris. Téléchargeable à l'adresse http://www.editionsdidier.com/files/media_file_14652.pdf

DJIKSRA, E., D. (1979) *A discipline of programming.* Prentice-Hall, Englewood Cliff.

GUY, M-N. (2013) *Utilisation du cadre théorique de la planification pour la conception d'algorithmes complexes par des élèves de lycée*. Mémoire de master de Didactique des mathématiques. Université Paris Diderot.
Téléchargeable à l'adresse http://dumas.ccsd.cnrs.fr/dumas-01090897

IREM FC (2010) Kaprekar.
http://www-irem.univfcomte.fr/index.php?id=article_13469_65_1&itemracine=9884&global_id_item=9884

LAGRANGE, J-B. (1992a) Représentations mentales des données informatiques et difficultés d'acquisition chez des débutants en programmation (première partie : hypothèses) *Enseignement Public et Informatique,* 67, 91-103.

http://www.epi.asso.fr/fic_pdf/b67p091.pdf

LAGRANGE, J-B. (1992b) Représentations mentales des données informatiques et difficultés d'acquisition chez des débutants en programmation (seconde partie : observations) *Enseignement Public et Informatique,* 68, 119-138.

http://www.epi.asso.fr/fic_pdf/b68p119.pdf

LAGRANGE, J-B. (2002) Les outils informatiques entre "sciences mathématiques" et enseignement. Une difficile transposition ? In Guin D., Trouche L. (Eds) *Intégrer les calculatrices symboliques : un problème didactique*. La Pensée Sauvage (Grenoble) 89-116.

LAVAL, D. (à paraître) L'algorithmique comme objet d'apprentissage de la démarche de preuve en théorie élémentaire des nombres : l'algorithme de Kaprekar. *Actes du Quatrième symposium international Espace de Travail Mathématique* (juillet 2014), Madrid, Espagne.

MODESTE, S. (2012) *Enseigner l'algorithme pour quoi ? Quelles nouvelles questions pour les mathématiques ? Quels apports pour l'apprentissage de la preuve ?* Thèse Université de Grenoble.

NGUYEN, C. T. (2005), *Étude didactique de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire à l'aide de la calculatrice.* Thèse de l'université Joseph Fourier, Grenoble.

ROGALSKI, M., SAMURÇAY, R. (1990) Acquisition of programming knowledge and skills. In J. M. Hoc, T. R. G. Green, R. Samurcay, David J. Gilmore (Eds)*Psychology of programming*. Academic Press.

SAMURÇAY, R. (1985) « Signification et fonctionnement du concept de variable informatique chez des élèves débutants », *Educational Studies in Mathem*atics, 16(2) 143-161.

SAMURCAY, R., ROUCHIER, A., (1990) Apprentissage de l'écriture et de l'interprétation des procédures récursives. *Recherches en didactique des Mathématiques.* 10 (2-3) 287-327.

SAMURÇAY, R., ROUCHIER, A. (1985) *De faire à faire faire. Planification d'actions dans la situation de programmation.* Enfance 2-3, 241-254.

ANNEXE. Énoncé des tâches données aux élèves

(À faire pour le 21 février)

On considère le programme de calcul suivant :

1. Choisir un nombre entier de trois chiffres.
2. Former le nombre obtenu en arrangeant les chiffres du nombre choisi en 1. dans l'ordre croissant.
3. Former le nombre obtenu en arrangeant les chiffres du nombre choisi en 1. dans l'ordre décroissant.
4. Calculer la différence des nombres obtenus en 2. et 3.
5. Recommencer (à partir de l'instruction 2.) avec le résultat obtenu en 4, jusqu'à obtenir un nombre déjà obtenu.

Tester à la main ce programme de calcul sur quelques exemples. Émettre une conjecture sur le résultat de cet algorithme.

Suggérer un test d'arrêt pour l'étape 5.

On envisage d'écrire l'algorithme correspondant à ce programme de calcul. Déterminer les variables nécessaires à l'écriture de cet algorithme.

(Séance du 21 février)

Quelles sont les étapes à programmer afin de pouvoir effectuer cet algorithme ?

Proposer un algorithme pour l'étape traitée par votre groupe.

(À faire pour le 21 mars)

Construire un algorithme effectuant le programme de calcul présenté dans la partie 1. Par exemple,

si on entre le nombre 392, on veut que l'algorithme renvoie :
$932 - 239 = 693$ $963 - 369 = 594$ $954 - 459 = 495$

(Séance du 21/03)

Programmer cet algorithme sur Algobox.