Evaluation de l'Algorithmique au lycée

version 1.9 Juin 2012
IREM Clermont Fd

Les retours des collègues confrontés à l'enseignement de l'algorithmique dans les classes de lycée, et la lecture des activités produites, mettent en évidence des difficultés à évaluer cette partie dans le cadre de l'enseignement des mathématiques.
Que doit-on évaluer ? Comment l'évaluer ?
Comment rendre cette évaluation constructive pour l'élève ? Comment placer cette évaluation au sein d'une évaluation de mathématiques ?...
Cette liste de questions n'est pas exhaustive et nous n'avons pas la prétention d'apporter une réponse tranchée à chacune d'entre elles. Nous exposons dans ce chapitre quelques éléments d'évaluation pouvant intervenir dans des séquences pédagogiques autour de l'algorithmique.
Nous pensons important de mettre en relief plusieurs axes à développer, axes avec des dépendances fortes.

Une évaluation de l'algorithmique au lycée :
- *Quoi ?*
- *Comment ?*
- *Objectif et place dans l'enseignement de mathématiques ?*

Nous allons essayer d'apporter quelques premiers éléments de réflexion :

1 Quoi ?

La réponse à cette première question va fortement influencer le contenu des autres réponses et il semble acquis auprès des collègues que les quatre capacités suivantes doivent être ciblées par une évaluation :

✓ **Lire un algorithme**

Être capable de lire un algorithme nécessite de reconnaître les concepts entrant en jeu dans son écriture. Ces notions sont clairement signalées

dans le texte du programme officiel, et entrent dans le cadre d'une progression établie sur l'année de seconde. Elles sont ensuite confortées dans le cycle terminal. Ce point devra donc faire l'objet d'une évaluation permanente, et non limitée au début de l'apprentissage.

✓ Exécuter un algorithme

Ce point vient en complément du précédent. C'est aussi une étape essentielle dans l'appropriation d'un algorithme. Pour s'assurer d'une lecture correcte d'un algorithme, l'exécution « à la main » de celui-ci est un indicateur pertinent.
Devant un algorithme difficile à appréhender, nous avons tous ressenti le besoin de le « faire tourner à la main »
Notre expérience, en tant qu'enseignant, nous permet de mettre en place cette stratégie de façon naturelle. Ce point va s'imposer beaucoup moins naturellement aux élèves, c'est pourquoi il apparaît indispensable de renforcer cette pratique au travers de l'évaluation afin de leur transmettre cette habitude.

✓ Comprendre un algorithme

Lire et exécuter un algorithme est possible, sans forcément en comprendre le sens et la fonction.
« Pourquoi avoir choisi cette structure ? », « Quel est le rôle de cette instruction ? »,« À quel problème répond cet algorithme ? », etc...
Nous avons là des questions qui ne relèvent plus du simple cadre de la lecture ou de l'exécution d'un algorithme mais qui en demandent une analyse plus critique.
L'évaluation de cette compétence permettra de mesurer la prise de conscience des élèves sur des problématiques au cœur de l'algorithmique, à savoir :

- **la correction d'un algorithme**
  Il n'est évidemment pas raisonnable de présenter des preuves de correction d'algorithmes aux élèves. Cependant, sans utiliser ces outils, il est possible de relever des erreurs dans un algorithme. C'est un point important, sur lequel insister dans nos évaluations. À ce sujet, les questions « Pourquoi cet algorithme est-il faux ? » « Proposer un jeu d'essai

pour tester la correction de cet algorithme. » sont souvent plus riches d'enseignement que la question « Pourquoi cet algorithme est-il correct ? ».

– **la terminaison d'un algorithme**
  Ici encore, sans faire de développement de cette notion, il est intéressant que les élèves se posent ce genre de questions : « L'exécution de cet algorithme se termine-t-elle ? », « Quel argument nous permet de penser que cet algorithme va rendre un résultat ? » etc ...
  Ces questions sont d'autant plus intéressantes, que, comme nous le verrons, elles vont pouvoir s'appuyer sur des notions mathématiques enseignées dans nos classes.
– **l'efficacité d'un algorithme**
  Nous utilisons volontairement ce terme plutôt que celui de complexité d'un algorithme. Il n'est pas question dans les classes de lycée d'entrer dans un développement de la notion de complexité d'un algorithme, mais les élèves peuvent déjà prendre conscience, que si deux algorithmes effectuent la même tâche, ils ne le font pas forcément de la même façon. Des questions peuvent être posées sur le nombre d'instructions exécutées par un algorithme, sur la nécessité d'un test donné, etc ...
  Nous utiliserons donc le terme d'efficacité afin d'éviter des confusions éventuelles avec la notion de complexité d'un algorithme.

La compréhension d'un algorithme peut se faire à différents niveaux. Pour illustrer ce point et comme exemple destiné aux enseignants, reprenons l'algorithme d'exponentiation rapide présenté au chapitre 2, paragraphe 1.3.6 :

```
Lire des entiers positifs a et b
res ← 1
tant que b > 0 faire
    si b impair alors
        res ← res × a
        b ← (b − 1)/2
    sinon
        b ← b/2
    a ← a × a
Retourner res
```

À moins de reconnaître cet algorithme, on peut se poser cette question : « Que fait cet algorithme ? »

On peut attendre plusieurs réponses, impliquant différents niveaux d'analyse :
- Une compréhension mot-à-mot
- L'algorithme se termine-t-il ?
- Quel est l'objectif de algorithme ?
- Comment fonctionne cet algorithme ?
- Pourquoi cet algorithme est-il correct ?
- Quelle est l'utilité de cet algorithme ?
  La réponse dépend, ici, du contexte d'utilisation de l'algorithme et fait intervenir des notions d'efficacité.

Le problème du niveau de réponse attendu va se poser et ne pourra être contrôlé qu'à partir de questions bien ciblées.
Les évaluations, mises en place aussi bien dans un cadre formatif que sommatif, devront tenir compte de cet élément.

✓ Concevoir un algorithme

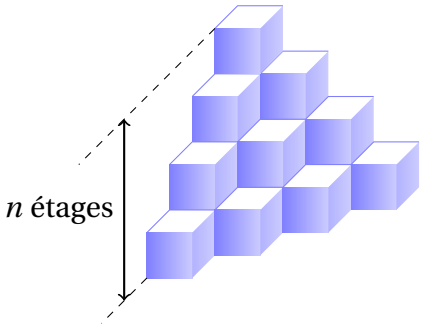
Les étapes précédentes sont un passage obligé avant de pouvoir écrire un algorithme.
En fin de terminale, nous pensons raisonnable d'être en mesure de demander à un élève l'écriture d'un algorithme simple relevant d'un problème classique.
Quelques exemples :
- « Écrire un algorithme retournant la somme des n premiers termes d'une suite donnée. »
- « Déterminer le plus petit entier naturel n, tel que $u_n > A$ avec A un réel donné et (u_n) une suite donnée, tendant vers $+\infty$. »
- « Être capable de concevoir un algorithme de recherche exhaustive dans une situation simple »
- « Recherche, sur un intervalle I, par dichotomie d'une valeur approchée, avec un précision donnée, de la solution de $f(x) = k$ avec f strictement monotone sur I »
- « Calcul d'une valeur approchée de l'intégrale par la méthode des rectangles »
- « Écrire un algorithme simulant la répétition d'une expérience aléatoire »
- « Simulation d'une marche aléatoire »

Au travers de ces exercices, les attendus sont les suivants :
- mettre en oeuvre une boucle `"POUR"`
- construction d'un test `"SI ... ALORS ... SINON"` avec un ou deux

niveaux d'imbrications
- créer une boucle `"TANT QUE"`
- commenter un algorithme
- implémenter un algorithme simple, lorsque c'est possible, sur une calculatrice ou un langage évolué

Si ces exercices semblent accessibles aux élèves, il peut en être autrement de l'écriture d'un algorithme plus complexe impliquant par exemple des structures conditionnelles ou itératives imbriquées, ou des variables jouant plusieurs rôles , comme dans cet exemple :

Écrire un algorithme, comportant une seule structure itérative, qui, pour un entier n donné, retourne le nombre de cubes de cette pyramide :



Une solution :

$a \leftarrow 0$
$b \leftarrow 0$
pour i **allant de** 1 **a** n
faire
    $a \leftarrow a + i$
    $b \leftarrow a + b$
retourner b

Dans ce cas, une évaluation ne peut se faire que dans un cadre très particulier :
- temps suffisant pour l'écriture. Ainsi, dans le cadre de certains algorithmes élaborés, cet exercice semble être exclu en temps limité ;
- maîtrise et recul suffisants par rapport aux concepts enseignés. Ce qui laisse entendre que ce type d'exercice, s'il est mis en place, ne peut l'être qu'en fin d'un cycle de formation ;

Cette forme d'exercice soulève des difficultés importantes. On peut, par contre, envisager des variantes plus adaptées à une évaluation en temps limité :
- algorithmes « à trous » à faire remplir par les élèves [1] ;
- réécriture d'un algorithme en imposant de changer de structure itérative ;
- réécriture d'un algorithme en imposant d'utiliser moins de structures conditionnelles ;

1. Ici, la frontière entre comprendre un algorithme et concevoir un algorithme devient difficile à percevoir.

– réécriture d'un algorithme en imposant d'utiliser moins de variables ;
– etc...

Nous illustrerons ces points par différents exemples dans la suite.

2 Comment évaluer ces compétences ?

Les compétences précédentes étant ciblées, nous pouvons présenter quelques exemples de stratégies permettant de les évaluer. Nous rappelons qu'il ne s'agit pas d'imposer une pratique, mais seulement de donner des pistes de réflexion, afin d'aider les collègues dans la construction de leur enseignement.

2.1 Lecture d'un algorithme

Concernant la lecture d'un algorithme, de simples exercices de reconnaissance pourront tout à fait convenir. On peut, dans un algorithme donné par exemple, faire reconnaître les instructions liées à des affectations, les blocs d'instructions liés à des structures conditionnelles ou des structures itératives.
Par exemple, la présentation d'un algorithme de ce genre :

```
lire n
c ← 0
pour k allant de 1 à n faire
    | si k divise n alors c ← c + 1
retourner c
```

peut donner lieu à ces questions :

- « *souligner les intructions liées à une affectation* »
- « *quelles sont les variables dans cet algorithme ?* »
- « *encadrer la (ou les) structures itératives.* »

  etc...

On peut aussi faire lire, et reconnaître, un même algorithme écrit sous différentes formes :
Parmi les algorithmes suivants, lesquels retournent le même résultat :

```
lire n
pour k allant de 1 à n faire
    | afficher k
```

Algorithme 1

```
saisir un entier n
afficher les valeurs d'un entier
variant de 1 à n
```

Algorithme 2

```
input n
for i=1 :n
  print i
```

Algorithme 3

À ce sujet, nous recommandons de familiariser les élèves à différentes écritures d'algorithmes :

- en français ;
- en pseudo-code de différents types ;
- sous formes de programmes, etc...

Ces questions vont constituer une partie importante de l'évaluation en début de formation mais ne seront pas à négliger non plus en cours et en fin de formation. Si elles n'apparaissent pas toujours dans les activités proposées par les enseignants, elles font bien souvent l'objet d'un traitement à l'oral. Un passage à l'écrit, cependant, ne nous semble pas superflu. Il permet en tout cas de poser un premier diagnostic fort utile.

2.2 Exécution d'un algorithme

Cette compétence est davantage ciblée que la précédente dans les évaluations écrites. Les formes retenues, le plus souvent, consistent à demander une exécution « à la main » d'un algorithme donné.
Par exemple, toujours à partir de l'algorithme :

```
lire n
c ← 0
pour k allant de 1 à n faire
    | si k divise n alors c ← c + 1
retourner c
```

On peut considérer la question suivante :

« Quelle valeur retourne cet algorithme, s'il est exécuté pour $n = 5$? »

On peut aussi considérer cet exemple :

« Exécuter l'algorithme suivant pour $x = 3$: »

Entrées : x un entier
Sorties : y un entier
si $x \geqslant 0$ **alors**
    $y \leftarrow x$
sinon
    $y \leftarrow -x$
retourner y

Deux réponses d'élèves ont alors été rencontrées :

Copie 1	**Copie 2**
Si $3 \geqslant 0$ alors	$3 \geqslant 0$
$y = 3$	alors j'affecte à y la valeur 3
sinon $y = -3$	Je retourne la valeur de y à savoir 3
Retourner 3	

Contrairement à l'élève 2, on peut supposer que l'élève 1 n'a pas compris la structure conditionnelle. L'exécution d'un algorithme est souvent liée à la compréhension de celui-ci.

2.3 Compréhension d'un algorithme

Il est possible d'exécuter pas à pas, un algorithme sans forcément en comprendre la fonction. C'est pourquoi, il nous apparaît nécessaire de bien distinguer ce point de celui relevant de la compréhension d'un algorithme.
On peut, par exemple, proposer un exercice dans lequel la compréhension d'un algorithme s'appuie sur son exécution :

Contexte de l'activité : en travail dirigé, établir, puis démontrer, une conjecture sur la parité du nombre de diviseurs d'un entier.

On se propose de mettre en œuvre un algorithme qui, pour un entier naturel n, non nul, choisi par l'utilisateur, affiche le nombre de diviseurs de n.
On donne quatre versions :

lire n
$c \leftarrow 0$
pour k **allant de** 1 **à** n **faire**
    si k *divise* n **alors** $c \leftarrow c+1$
afficher c

Algorithme 1

lire n
$c \leftarrow 2$
pour k **allant de** 2 **à** $n-1$ **faire**
    si k *divise* n **alors** $c \leftarrow c+1$
afficher c

Algorithme 2

lire n
$c \leftarrow 1$
pour k **allant de** 2 **à** n **faire**
    si k *divise* n **alors** $c \leftarrow c+1$
afficher c

Algorithme 3

lire n
$c \leftarrow 1$
pour k **allant de** 1 **à** $n-1$ **faire**
    si k *divise* n **alors** $c \leftarrow c+1$
afficher c

Algorithme 4

Parmi ces quatre versions, une ne répond pas au problème. Laquelle et pourquoi ?

L'exercice nécessite de vraiment se plonger dans la lecture et l'exécution des algorithmes. Les élèves, pour pouvoir répondre, vont devoir non seulement exécuter ces algorithmes mais aussi en faire une analyse critique.
Cette analyse critique peut être renforcée par des questions plus précises, comme celle-ci :
« Pour chacun de ces algorithmes, quelles sont les situations dans lesquelles on n'entre pas dans la boucle ? »
D'autre part, ce genre d'exercice a l'intérêt de présenter les choses sous forme de défi. Lors de cette séquence, on a pu constater que les élèves se prenaient réellement au jeu, mettant en place des stratégies différentes pour trouver la bonne réponse. Certains ont, ainsi, testé à la main différents jeux de valeurs, ou l'ont fait à partir de versions implémentées dans un langage de programmation et d'autres ont étudié la correction des algorithmes en identifiant les points critiques.
Toujours dans le cadre d'une évaluation en travail dirigé, il peut être intéressant de compléter cette étude par une discussion portant l'efficacité de l'algorithme :

Des trois versions correctes, laquelle semble la plus judicieuse [a] ? Justifier.

a. Il s'agit, ici, d'une question purement théorique car, dans la réalité, réaliser un ou deux tests inutiles n'a pas de conséquence

Il est alors amusant de constater que les élèves sont surpris par le fait que plusieurs réponses différentes, à partir du moment où elles sont argumentées, sont acceptables.
Les différentes réponses apportées ont été :

- Pour l'algorithme 1 :
  Plus facile à comprendre, à lire.
  Pour certains l'algorithme 1 est moins « tordu » donc plus fiable.
  Pour un autre groupe, toujours dans un soucis de fiabilité, l'algorithme 1 devait être choisi car il réalise tous les tests. Cela a permis de revenir sur les points aborbés dans la question 1.
- Pour les algorithmes 3 et 4 :
  Le choix a été dirigé par l'idée qu'ils permettent un plus petit nombre de tests. Certains élèves les ont choisis sans les différencier, et d'autres sont allés plus loin, en optant pour l'algorithme 4 qui réalise un dernier test de divisibilité sur un entier moins grand.
  Un groupe a fait remarquer qu'au final, cela ne changeait pas grand chose.

Les exercices du type « algorithmes à trous », permettent de s'assurer de la compréhension de points précis, comme par exemple le rôle d'une variable d'accumulation dans cet exercice :

Compléter l'algorithme suivant afin qu'il retourne la somme des entiers naturels de 0 à n, pour n donné :

lire n
$s \leftarrow 0$
pour k **allant de** 0 **à** n **faire**
    $s \leftarrow$.....
retourner s

Le choix des parties à compléter n'est pas laissé au hasard. À travers l'exemple suivant, on cible la compréhension de structure conditionnelle :

L'algorithme suivant est censé, à partir de deux nombres, tous deux non nuls, nous informer si leur produit est négatif ou positif.
Certaines parties ont été effacées, retrouver le texte manquant :

```
afficher "saisir votre premier nombre"
lire a
afficher "saisir votre second nombre"
lire b
si a > 0 alors
    si b > 0 alors
        afficher "le produit est ....."
    sinon
        afficher "le produit est ....."
sinon
    si b > 0 alors
        afficher "le produit est ....."
    sinon
        afficher "le produit est ....."
```

On peut aussi par exemple s'assurer de la compréhension d'une clause d'arrêt d'une boucle **tant que** :

Une balle élastique est lâchée d'une hauteur de 100 cm au-dessus d'une table ; elle rebondit plusieurs fois.
On appelle h_n la hauteur en centimètre du n^{eme} rebond, et h_0 vaut 100.
La hauteur atteinte à chaque rebond est égale à 9/10 de la hauteur du rebond précédent.
On se propose de déterminer expérimentalement à partir de quel rebond la hauteur deviendra inférieure à 1 cm.
Compléter l'algorithme ci-dessous pour pouvoir répondre au problème :

```
h ← 100
n ← 0
tant que ..... faire
    h reçoit h × 0,9
    n reçoit n + 1
afficher la valeur de n
```

Toujours à partir du même exemple, on aurait pu cibler un autre point portant sur la génération des termes d'une suite géométrique :

h reçoit 100
n reçoit 0
tant que $h \geq 1$ **faire**
    $h \leftarrow h \times$ .....
    $n \leftarrow$ .....
afficher la valeur de n

Par contre, il n'est peut-être pas utile de cibler simultanément les deux points, à moins de vouloir travailler une compétence portant davantage sur la conception d'un algorithme.
Un autre type d'exercice intéressant consiste à demander aux élèves de corriger un algorithme comportant une erreur.
Ce qui peut donner :

Corriger l'algorithme suivant afin qu'il retourne la somme des entiers naturels de 0 à n, pour n donné :

lire n
$s \leftarrow 0$
pour k **allant de** 0 **à** n **faire**
    $s \leftarrow s+1$
écrire s

La difficulté supplémentaire est ici de trouver l'erreur pour pouvoir la corriger. Il est intéressant dans ce type d'exercice de choisir des erreurs souvent relevées dans les copies. Il s'agira tout de même de rester raisonnable quant à la difficulté des corrections demandées.
Ce type d'exercice permet aussi d'insister sur certains points. Dans l'exemple suivant, on illustre les différences entre deux structures itératives :

On souhaite afficher la liste des 2012 premiers nombres premiers. Pour cela on dispose d'une fonction `premier` fonctionnant de la façon suivante : Pour n entier naturel, `premier(`n`)` retourne `vrai` si n est premier et `faux` dans le cas contraire.
On propose l'algorithme suivant :

pour k **allant de** *1* **à** 2012 **faire**
  si *premier(k)* **alors**
    afficher k

Expliquer en quoi la structure itérative choisie, ne permet pas de répondre au problème.
Quelle structure itérative semble plus adaptée ?

Les exercices précédents sont aussi une bonne préparation à la conception d'algorithmes.
Soulignons que le manque de recul dans l'enseignement de l'algorithmique au lycée et la frontière, souvent mince, entre les mathématiques et l'algorithmique dans les activités proposées, vont engendrer des difficultés supplémentaires dans l'évaluation des algorithmes écrits par les élèves. Face à des réponses erronées, il n'est pas toujours aisé de cibler le problème. L'erreur est-elle engendrée par des difficultés de compréhension en mathématiques ou [2] en algorithmique ?
Deux exemples, relevés dans des copies :

Contexte : on demande un algorithme permettant de déterminer le plus petit rang n pour lequel $u_n \leq 65$*, où* (u_n) *est la suite définie par* $u_0 = 100$ *et* $u_{n+1} = \frac{1}{2}u_n + 30$.

Variables
n est du type nombre
u est du type nombre
Début algorithme
n prend la valeur 0
u prend la valeur 100
si $u > 65$ **alors**
  u prend la valeur $\frac{1}{2}u + 30$
  n prend la valeur $n + 1$
Afficher n

2. « ou » inclusif, bien sûr !

Variables
n est du type nombre
u_n est du type nombre
Début algorithme
n prend la valeur 0
u_n prend la valeur $60+40\times\left(\frac{1}{2}\right)^n$ [a]
Si $u_n > 65$ **alors** u_n prend la valeur $60+40\times\left(\frac{1}{2}\right)^n$
Afficher n

a. cette expression est le résultat de l'une des questions de l'exercice.

Remarquons que la structure itérative attendue n'est mise en place dans aucun de ces algorithmes.
Face à ces productions, la première réaction a été de penser que les élèves ne maîtrisaient pas ces structures et donc que le problème ne relevait que de l'algorithmique. Après avoir intérrogé les élèves, il s'est avéré que le problème était aussi lié à la façon d'appréhender les suites définies par récurrence.
La structure itérative à mettre en place ici, est fortement liée à la définition de la suite. Ainsi, des difficultés en mathématiques vont se traduire immédiatement par de mauvaises traductions algorithmiques. Cet exemple illustre la difficulté à construire une évaluation d'algorithmique non biaisée par des problèmes de compréhension en mathématiques.
On présente ci-dessous quelques stratégies souvent utilisées en situation :
Demander à un élève ce que produit l'algorithme, si on remplace une instruction par une autre, ce qui lui permet de prendre conscience du rôle et de la portée de cette même instruction.

Par exemple, considérons la méthode de Monte-Carlo appliquée à l'estimation du nombre de triplets (a,b,c), de l'ensemble $\mathscr{E} = \{0,1,2,\ldots,9\}$, vérifiant $a<b<c$:

$S \leftarrow 0$
pour k **allant de** 1 **à** 10^4 **faire**
  $a \leftarrow$ entier aléatoire compris entre 0 et 9
  $b \leftarrow$ entier aléatoire compris entre 0 et 9
  $c \leftarrow$ entier aléatoire compris entre 0 et 9
  si $a < b$ *et* $b < c$ **alors** $S \leftarrow S + 1$
retourner $\dfrac{S}{10^4} \times 10^3$

Les questions, que l'on peut tirer d'un tel algorithme sont très nombreuses :

- *« Que se passe-t-il si on remplace la ligne*
  si $a < b$ *et* $b < c$ **alors** $S \leftarrow S + 1$
  par
  $S \leftarrow S + 1$ *? »*
- *« Que se passe-t-il si on modifie l'ordre des instructions de la façon suivante : »*

ou encore :

- Quelles sont les valeurs que peut retourner l'algorithme suivant :

$S \leftarrow 0$
$a \leftarrow$ entier aléatoire compris entre 0 et 9
$b \leftarrow$ entier aléatoire compris entre 0 et 9
$c \leftarrow$ entier aléatoire compris entre 0 et 9
pour k **allant de** 1 **à** 10^4 **faire**
  si $a < b$ *et* $b < c$ **alors** $S \leftarrow S + 1$
retourner $\dfrac{S}{10^4} \times 10^3$

Plus en rapport avec le programme de mathématiques, on peut aussi poser cette question :

Cet algorithme

$S \leftarrow 0$
pour k **allant de** 1 **à** 10 **faire**
  $a \leftarrow$ entier aléatoire compris entre 0 et 9
  $b \leftarrow$ entier aléatoire compris entre 0 et 9
  $c \leftarrow$ entier aléatoire compris entre 0 et 9
  si $a < b$ *et* $b < c$ **alors** $S \leftarrow S + 1$
retourner $\dfrac{S}{10^4} \times 10^3$

ne permet pas une estimation acceptable du nombre recherché, que peut-on modifier pour remédier à ce problème ?

En formulant le problème de cette façon, on obtient une question qui relève à la fois du cadre du programme de mathématiques et de celui d'algorithmique. Si on souhaite ne faire porter la question que sur le test, en se débarassant du problème mathématique, on peut alors demander une réécriture de l'algorithme en utilisant des structures conditionnelles imbriquées, ou inversement. L'objectif étant de passer de l'une de ces deux versions à l'autre :

$S \leftarrow 0$
pour k **allant de** 1 **à** 10^4 **faire**
  $a \leftarrow$ entier aléatoire compris entre 0 et 9
  $b \leftarrow$ entier aléatoire compris entre 0 et 9
  $c \leftarrow$ entier aléatoire compris entre 0 et 9
  si $a < b$ **alors**
    si $b < c$ **alors**
      $S \leftarrow S + 1$
retourner $\dfrac{S}{10^4} \times 10^3$

$S \leftarrow 0$
pour k **allant de** 1 **à** 10^4 **faire**
  $a \leftarrow$ entier aléatoire compris entre 0 et 9
  $b \leftarrow$ entier aléatoire compris entre 0 et 9
  $c \leftarrow$ entier aléatoire compris entre 0 et 9
  si $a < b$ *et* $b < c$ **alors**
    $S \leftarrow S + 1$
retourner $\dfrac{S}{10^4} \times 10^3$

Demander de modifier un algorithme existant pour répondre à une autre problématique, est un exercice intéressant. À titre d'exemple, on peut considérer l'exercice suivant :

On considère la fonction f définie sur $[-5;15]$ par :

$$f(x) = -0,05x^3 + x^2 - x + 1.$$

L'algorithme suivant retourne l'entier $k \in [-5;15]$ pour lequel l'image par la fonction f est la plus petite possible :

$m \leftarrow -5$
pour k **allant de** -4 **à** 15 **faire**
    si $f(k) < f(m)$ **alors** $m \leftarrow k$
écrire m

Modifier l'algorithme précédent afin qu'il retourne l'entier $k \in [-5;15]$ pour lequel l'image par la fonction f est la plus **grande** possible.

Dans les exemples précédents, les évaluations portent sur la compréhension d'algorithmes pré-existant. On évalue alors une capacité d'analyse mais pas encore créative. Au-delà de la compréhension des notions d'algorithmique va cependant s'ajouter une difficulté supplémentaire. Ce type d'exercice nécessite, en effet, de comprendre la démarche du concepteur de l'algorithme. En général, elle n'est pas unique et peut souvent être éloignée de la démarche naturelle de l'élève. C'est une difficulté non négligeable dont il convient de tenir compte dans la mise en place d'une évaluation.
D'autre part le fait de demander à un élève de corriger, ou compléter, un algorithme, dépasse déjà en pratique le cadre d'une simple capacité d'analyse ce qui fait de ce type d'exercice est une étape préalable au passage à la conception d'un algorithme, cœur de l'algorithmique.

2.4 Conception d'un algorithme

Dans l'exercice de conception d'un algorithme, la difficulté réside non seulement dans la décomposition d'une tâche complexe en tâches élémentaires mais aussi dans l'écriture codifiée de celles-ci. Cependant ce type de démarche va laisser davantage le champ libre aux élèves, et il est intéressant d'en tenir compte dans l'analyse des réponses produites et des choix effectués par les élèves, même si cela n'est pas toujours aisé.
Pour illustrer ce point, nous pouvons considérer l'exemple suivant :

Écrire un algorithme retournant le plus petit entier n tel que $u_n > 50$ avec (u_n) la suite définie par $u_0 = 1$ et $u_{n+1} = \frac{3}{2}u_n + \frac{1}{3}$.

pour lequel un élève répond :

Variables
n est du type nombre
u est du type nombre
F est du type nombre
Début algorithme
lire F `# il faut choisir` F `très grand pour être sûre`
n prend la valeur 0
u prend la valeur 1
pour k **allant de** 1 **à** F **faire**
    $u \leftarrow \frac{3}{2}u + \frac{1}{3}$
    si $u \leq 50$ **alors** n prend la valeur $n+1$
Afficher n

Si on passe sur l'erreur [3] du test non effectué au premier rang, on peut constater que le choix de la structure itérative la plus adaptée n'a pas été évidente pour l'élève. À la vue du commentaire, un problème a été perçu par l'élève mais il ne voit pas comment le traiter sous forme algorithmique. On peut remarquer de plus que, dans ce cas précis, l'utilisation d'une boucle **Pour** lui a vraisemblablement beaucoup compliqué la tâche.
Ce point n'aurait pas été mis en évidence, si une certaine liberté n'avait pas été laissée à l'élève.
On peut considérer l'exemple suivant :

Écrire un algorithme de résolution d'une équation du second degré à partir de ses coefficients donnés.

et se demander comment réagir face à la réponse :

3. A-t-on le droit de considérer que c'est une erreur ? Ici, la définition de u_n est intrinsèque à l'algorithme, et le fait de savoir que $u_0 = 1 < 50$, nous dispense de ce test.

#a, b, c et Δ désignent des réels
Lire a, b et c
Δ reçoit $b^2 - 4ac$
si $\Delta < 0$ **alors**
    afficher « l'équation n'admet pas de solution réelle »
si $\Delta = 0$ **alors**
    x_0 reçoit $-\dfrac{b}{2a}$
    afficher « l'équation admet une solution : »
    afficher x_0
si $\Delta > 0$ **alors**
    x_1 reçoit $\dfrac{-b-\sqrt{\Delta}}{2a}$
    x_2 reçoit $\dfrac{-b+\sqrt{\Delta}}{2a}$
    afficher « l'équation admet deux solutions : »
    afficher x_1 et x_2

Dans sa réponse, l'élève a choisi de faire séparer chacun des tests sur le signe de Δ dans des structures conditionnelles distinctes. Nous savons que si Δ est strictement négatif, ce test sera effectué inutilement deux fois mais l'algorithme reste correct.
Plusieurs interprétations peuvent être tirées de cette réponse :

- l'élève ne maîtrise pas suffisamment les structures conditionnelles imbriquées
- l'élève s'est contenté de cette réponse car l'algorithme répond au problème demandé

Après avoir questionné l'élève, il s'est avéré que ce choix était délibéré et motivé par le seul soucis de lisibilité de l'algorithme. Cet argument justifie tout à fait ce choix.
Le maniement des structures conditionnelles imbriquées n'est pas identique en algorithmique et en mathématiques, ce qui peut conduire à l'écriture d'algorithmes erronés.

Par exemple, la fonction f définie sur $\mathbb{R}$ par :

$$f(x) = \begin{cases} 3x-2 \text{ si } x<1 \\ x^2 \text{ si } x \geq 1 \text{ et } x \leq 2 \\ 2x \text{ si } x>2 \end{cases}$$

est, quelquefois réécrite par les élèves sous cette forme :

si $x<1$ alors $f(x)=3x-2$
si $x \geq 1$ et $x \leq 2$ alors $f(x)=x^2$ sinon $f(x)=2x$

Le dernier « sinon » porte implicitement aussi sur la clause « $x<1$ ». Cette définition parfaitement compréhensible en Français, est incorrecte du point de vue algorithmique, et conduit à ce genre d'algorithme (observé dans des copies d'élèves) :

```
lire x
si x < 1 alors résultat ← 3x − 2
si x ≥ 1 et x ≤ 2 alors
    résultat ← x²
sinon
    résultat ← 2x
afficher résultat
```

L'exécution de l'algorithme, pour des valeurs de x inférieures à 1, va conduire à des résultats erronés.
Les affectations multiples dans un algorithme soulèvent aussi des problèmes qui sont traités de façon transparente en mathématiques. On peut illustrer ce propos en considérant cet exercice (Antilles S 2004) :

On définit les suites (a_n) et (b_n) par $a_0 = 1$, $b_0 = 7$ et
$$\begin{cases} a_{n+1} &= \dfrac{1}{3}(2a_n + b_n) \\ b_{n+1} &= \dfrac{1}{3}(a_n + 2b_n) \end{cases}$$
Soit D une droite munie d'un repère $\left(\text{O} ; \vec{\text{i}}\right)$.
Pour tout $n \in \mathbb{N}$, on considère les points A_n et B_n d'abscisses respectives a_n et b_n.

1. Placez les points A_0, B_0, A_1, B_1, A_2 et B_2.
2. Soit (u_n) la suite définie par $u_n = b_n - a_n$ pour tout $n \in \mathbb{N}$. Démontrez que (u_n) est une suite géométrique dont on précisera la raison et le premier terme.
   Exprimez u_n en fonction de n. ...

Son adaptation, dans le cadre de l'algorithmique, peut amener les élèves à élaborer un algorithme permettant le calcul des termes a_n et b_n pour n donné.
Face à cette production :

```
...
tant que k < n faire
    a ← (2a + b)/3
    b ← (a + 2b)/3
    k ← k + 1
...
```

il est facile d'analyser l'erreur produite, et d'envisager des remédiations, comme par exemple l'éxécution de l'algorithme pas à pas, en relevant l'état des variables à chaque instruction.
Mais comment réagir face à un algorithme juste, comme dans ces deux versions :

...
tant que $k < n$ **faire**
    $a, b \leftarrow \dfrac{2a+b}{3}, \dfrac{a+2b}{3}$
    $k \leftarrow k+1$
...

...
tant que $k < n$ **faire**
    $temp \leftarrow a$
    $a \leftarrow \dfrac{2a+b}{3}$
    $b \leftarrow \dfrac{temp+2b}{3}$
    $k \leftarrow k+1$
...

La première version est beaucoup plus facile à lire que la seconde mais elle laisse de côté une partie du problème d'affectation multiple, ou la considère comme déja résolue.
Doit-on alors accepter les deux versions, ou seulement l'une des deux ?
Même s'il nous semble important de priviligier la seconde version dans nos enseignements, nous pensons important de valider les deux versions dans le cadre d'une évaluation. En effet si on accepte l'idée d'affectation multiple [4] à travers le symbole « ← » dans nos algorithmes, le premier algorithme doit être validé. Il s'agit seulement alors de s'assurer que le problème de l'affectation multiple dans nos deux suites a bien été perçu par les élèves.
Nous n'avons pas encore une pratique et un recul important dans l'enseignement de l'algorithmique au lycée. C'est pourquoi il convient d'être très prudent quant aux conclusions que l'on peut tirer de la réponse écrite d'un élève. Cependant les indications que l'on peut tirer de certaines réponses peuvent cibler des difficultés bien précises. C'est pourquoi, même si l'exercice d'écriture d'un algorithme est difficile à donner dans une épreuve en temps limité, il est important de ne pas l'occulter.
Lors de l'écriture d'un algorithme, les élèves vont être confrontés au problème de la mise en forme. Les recommandations dans les programmes du lycée nous incitent à éviter tout excès de formalisme, ce qui nous apparaît une bonne chose. Il est important de garder une liberté d'écriture. La mise en forme d'un algorithme ne doit pas être un frein pour l'élève à l'élaboration d'une solution d'un problème mathématique mais plutôt l'occasion de proposer une réponse dans un langage qui lui est plus naturel ou en tout cas plus accessible. Il apparaît ainsi essentiel d'accepter différentes formes d'écriture de nos algorithmes.

4. Certains langages, comme Python, autorisent ce type d'affectation

3 Intégration de l'évaluation de l'algorithmique dans celle des mathématiques

Au travers de l'algorithmique, nous avons l'occasion de présenter, et d'évaluer, des points importants en mathématiques, et que l'on n'avait pas l'habitude, ou des difficultés, à illustrer.
Un exemple porte sur la nature même des problèmes traités par l'algorithmique. Les algorithmes sont liés à des ensembles discrets et finis de valeurs et, dans certains cas, l'apport des mathématiques va nous permettre de dépasser ce cadre.
Considérons ce problème :

On se propose de résoudre, dans $\mathbb{N}^3$, l'équation $a^2+b^2+c^2=2386$ avec $a \leq b \leq c$.

1. Montrer que $a^2+b^2+c^2 \geq a^2$ et en déduire que $a<17$.
2. Justifier que $c^2<2386$ et en déduire que $c<49$.
3. Concevoir un algorithme exhaustif qui retourne tous les triplets d'entiers $(a;b;c)$ solutions de l'équation $a^2+b^2+c^2=2386$ avec $a \leq b \leq c$.

On peut compléter l'exercice par une question un peu plus ouverte :

4. Comment peut-on diminuer le nombre d'itérations dans l'algorithme précédent ?

Cet exercice repose sur le principe suivant :
On ne peut écrire un algorithme de recherche des solutions dans $\mathbb{N}^3$ tout entier.
On peut par contre se débarasser du problème d'infini, en montrant que les solutions de l'équation appartiennent à un ensemble borné.
À partir de $a \leq b \leq c$, on montre facilement que $3a^2 \leq a^2+b^2+c^2$
Par conséquent, pour $3a^2>2386$, c'est-à-dire $a \geq 17$, aucun triplet $(a;b;c)$ n'est solution.
De même, $a^2+b^2+c^2 \geq c^2$ donc pour $c^2>2386$, c'est-à-dire $c \geq 49$, aucun triplet $(a;b;c)$ n'est solution.
On a ainsi $0 \leq a \leq 16$ et $0 \leq c \leq 48$ sachant que $a \leq b \leq c$.
Il ne reste plus qu'un nombre fini de valeurs à explorer, ce qui peut se traiter à l'aide d'un algorithme exhaustif :

```
pour a allant de 0 à 16 faire
    pour b allant de a à 48 faire
        pour c allant de b à 48 faire
            si a² + b² + c² = 2 386 alors afficher (a; b; c)
```

Dans ce type d'exercice, l'algorithme devient un élément de preuve du problème mathématique posé. C'est un emploi intéressant de l'algorithmique dans nos évaluations mathématiques. En proposant des exemples allant dans ce sens, on peut espérer favoriser chez les élèves une démarche visant à résoudre des problèmes mathématiques posés sous une forme plus ouverte comme celui-ci par exemple :

> Réaliser une recherche exhaustive de tous les couples d'entiers naturels $(x; y)$ solutions de l'inéquation : $5x^2 - 4xy + y^2 \leq 100$.

Dont une solution peut être :

Pour tous les réels x et y, on a :

$$\begin{aligned} 5x^2 - 4xy + y^2 &= x^2 + 4x^2 - 4xy + y^2 \\ &= x^2 + (2x - y)^2 \end{aligned}$$

On constate que :
si $x^2 > 100$ ou $(2x-y)^2 > 100$ alors l'inéquation ne peut être vérifiée.
La première inéquation impose d'avoir

$$-10 \leq x \leq 10.$$

La seconde inéquation impose d'avoir

$$-10 \leq 2x - y \leq 10.$$

ce dernier résultat donne $y \leq 2x + 10$, puis $y \leq 30$.
Nous avons ainsi trouvé un majorant des solutions, on peut alors établir la liste des solutions à partir d'un algorithme exhaustif :

pour x **allant de** 0 **à** 10 **faire**
  pour y **allant de** 0 **à** 30 **faire**
    si $5x^2 - 4xy + y^2 \leq 100$ **alors** *afficher* $(x; y)$

On peut remarquer, à ce sujet, qu'il est bien plus intéressant, et important, de fournir l'algorithme de recherche plutôt que l'ensemble des solutions lui-même.

De la même façon, ce sont des résultats mathématiques qui nous permettent de valider certains algorithmes et nos élèves peuvent déjà en prendre conscience.
Toujours dans l'esprit de questions plus ouvertes, on peut imaginer ce problème :

On dispose d'une urne contenant 1 000 boules numérotées de 1 à 1 000.
On tire simultanément deux boules de l'urne.
Déterminer la probabilité que les numéros des boules tirées soient deux entiers premiers entre eux.
On pourra s'aider d'un algorithme utilisant une fonction `pgcd`*, qui renvoie le pgcd de deux entiers.*

Dont une solution possible est :

Le nombre possible de couple d'entiers formés par le tirage des deux boules, nous est donné par :

$$\binom{1000}{2} = 499\,500$$

Le nombre de couples formés par des entiers premiers entre eux nous est donné par l'algorithme :

$S \leftarrow 0$
pour k **allant de** 1 **à** 1 000 **faire**
  pour j **allant de** $k+1$ **à** 1 000 **faire**
    si $pgcd(k, j) = 1$ **alors** $S \leftarrow S+1$
écrire S

La probabilité cherchée est alors égale à la valeur de S retournée par l'algorithme divisée par 499 500.

On peut remarquer qu'une réponse peut aussi être donnée par l'algorithme suivant :

$S \leftarrow 0$
$N \leftarrow 0$
pour k **allant de** 1 **à** 1 000 **faire**
  pour j **allant de** $k+1$ **à** 1 000 **faire**
    $N \leftarrow N+1$
    si $pgcd(k, j) = 1$ **alors** $S \leftarrow S+1$
écrire $\dfrac{S}{N}$

Les notions, présentées au lycée, sur les suites convergentes fournissent des exemples intéressants traitant de la terminaison d'un algorithme. On peut présenter les choses ainsi :

D'après Baccalauréat S Nouvelle-Calédonie mars 2011

Soit (u_n) la suite définie par $\begin{cases} u_0 &= 1 \\ u_{n+1} &= u_n - \ln\left(u_n^2+1\right) \text{ pour } n \in \mathbb{N}. \end{cases}$

Partie A

Soit f la fonction définie sur $\mathbb{R}$ par $f(x) = x - \ln\left(x^2+1\right)$.

1. Résoudre dans $\mathbb{R}$ l'équation $f(x) = x$.
2. Étudier le sens de variation de la fonction f sur l'intervalle $[0\ ;\ 1]$.
   En déduire que si $x \in [0\ ;\ 1]$ alors $f(x) \in [0\ ;\ 1]$.

Partie B

1. Démontrer par récurrence que, pour $n \geqslant 0,\ u_n \in [0\ ;\ 1]$.
2. Étudier le sens de variation de la suite (u_n).
3. Démontrer que la suite (u_n) converge vers 0.
4. On propose l'algorithme suivant :

```
u ← 1
k ← 0
tant que u > 0,05 faire
    k ← k + 1
    u ← u − ln(u² + 1)
retourner k
```

   (a) Que retourne cet algorithme à la fin de son exécution ?

   (b) Quel résultat mathématique, démontré précédemment, nous permet d'affirmer que l'exécution de cet algorithme se termine.

Dans la dernière question, nous avons une application concrète de la convergence d'une suite. Sans ce résultat, sur la suite (u_n), nous n'avons aucun moyen de nous assurer de la terminaison, et donc de la validité, de cet algorithme.
On peut très bien imaginer de construire aussi l'évaluation directement autour de l'algorithme, et proposer des questions nécessitant un recours plus fort aux mathématiques :

On considère la fonction f définie sur $]0;+\infty[$ par

$$f(x) = \frac{x}{\ln(1+x)}.$$

Justifier que l'algorithme suivant se termine :

lire a % a désigne un nombre réel
$k \leftarrow 1$
tant que $f(k) < a$ **faire**
    $k \leftarrow k+1$
retourner k

Toujours pour pousser les élèves dans cette réflexion, pourquoi ne pas leur proposer de débattre [5] sur la validité de cet algorithme :

D'après Pondichéry Terminale S - avril 2012-

Variables	a, b, c, d, e sont des variables du type entier
Initialisation	$a := 0$; $b := 0$; $c := 0$; $d := 0$; $e := 0$
Traitement	Tant que $(a = b)$ ou $(a = c)$ ou $(a = d)$ ou $(a = e)$ ou $(b = c)$ ou $(b = d)$ ou $(b = e)$ ou $(c = d)$ ou $(c = e)$ ou $(d = e)$ Début du tant que $a :=$ rand(1, 50) ; $b :=$ rand(1, 50) ; $c :=$ rand(1, 50) ; $d :=$ rand(1, 50) ; $e :=$ rand(1, 50) Fin du tant que
Sortie	Afficher a, b, c, d, e

Pour pousser le vice, on peut faire programmer et exécuter cet algorithme. L'exécution de ce programme va bien sûr toujours se terminer alors que l'algorithme pose un problème du point de vue de sa terminaison. Cet algorithme, dont on peut imaginer des versions plus simples encore, permet de soulever d'intéressantes questions auprès des élèves.

5. Il ne s'agit en aucun cas d'utiliser cette question dans le cadre d'une évaluation sommative.

4 Pour conclure

Nous espérons avoir montré qu'en choisissant des questions pertinentes, on bénéficie d'un outil de diagnostic utile, non seulement dans le cadre de l'enseignement de l'algorithmique mais aussi dans celui des mathématiques.
Cependant la mise en place des concepts d'algorithmique, pour être efficace, suppose que les notions mathématiques intervenant soient bien perçues par les élèves. Ainsi, dans le cadre d'une évaluation sommative de l'algorithmique, il est sans doute préférable que les notions de mathématiques servant de support soient faciles d'accès. Le risque est, en effet, important de voir des réponses d'élèves biaisées par des difficultés en mathématiques. Il reste, cependant, beaucoup de situations dans lesquelles cette évaluation ne pourra être que conjointe à une évaluation en mathématiques.

Par ailleurs, ces notions d'algorithmique ne sont pas faciles à appréhender par les élèves. C'est pourquoi si on ne veut pas voir cet enseignement se retrouver avec une connotation élitiste, il conviendra d'être des plus raisonnables quant à la difficulté des questions posées. Enfin, il nous apparaît important de soulever un dernier point concernant la rédaction de ces évaluations. L'algorithmique est une matière nouvelle dans nos enseignements, et en cela les évaluations proposées ne peuvent se satisfaire d'implicites. Nos évaluations, en mathématiques, présentent dans leur formulation beaucoup d'implicites qui sont le fruit des pratiques et de la culture commune de nos enseignements. Dès lors, certains attendus n'ont pas à être détaillés explicitement sans que cela perturbe les élèves. Il en va tout autrement en algorithmique, où la plus grande vigilance va s'avérer nécessaire dans la mise en forme d'une évaluation. Ces implicites, ainsi qu'une culture commune, vont progressivement se mettre en place dans nos enseignements. Ils seront probablement largement influencés par la forme des évaluations que l'on proposera aux élèves.