Enseigner l'algorithmique au lycée

Les bases et méthodes

Les difficultés rencontrées et les solutions

Préambule / Licence

Le document est partagé sous la licence CC-BY-NC-ND : l'œuvre peut être librement utilisée, à la condition de l'attribuer à l'auteur en citant son nom, aucune utilisation commerciale n'est autorisée et le document doit être partagé à l'identique.

Si vous souhaitez utiliser ce document en cours (en classe ou en cours particuliers) sans le filigrane merci de nous faire parvenir un mail à blog@partiprof.fr indiquant les finalités de votre utilisation du document. Nous vous ferons alors parvenir une version non filigranée.

Sommaire

Préambule / Licence 1
I. Introduction et présentation 2
II. Définition / But 2
III. Notation / Légende 2
IV. Le pseudo-code 3
V. Le texte 3
VI. Les variables 3
VII. Initialisation des variables 4
VIII. Les entrées utilisateurs 4
IX. Travail sur les variables 5
X. Les différentes parties de l'algorithme 6
XI. Les boucles 6
1. La boucle "Pour" 6
2. La boucle "Tant que" 9
XII. Les conditions 10
1. Les conditions simples 10
2. Les conditions multiples 11
XIII. La géométrie 11
XIV. Nombres aléatoires 11
XV. Quelques exemples 12
1. Niveau seconde 12
2. Niveau première S 13
3. Niveau terminale S 13
XVI. Problèmes courants 15
1. Noms de variables 15
2. Les boucles 15
3. Les "meilleurs" 15
XVII. Conclusions 16

I. Introduction et présentation

Depuis 2012 l'algorithmique a fait son entrée dans les programmes ainsi que dans les sujets du bac de différentes filières. Que ce soit par manque de temps ou par manque d'envie (rien ne dit qu'un prof de maths soit apte à enseigner l'algorithmique) cette partie du programme est souvent négligée ou mal expliquée. On trouve même des sujets d'examens avec des algorithmes faux ! Ce document a donc pour but de reprendre les notions de bases nécessaires à la compréhension de l'algorithmique tel qu'on l'entend au lycée.

Pour ma part j'ai enseigné l'algorithmique au lycée et à des élèves en cours particuliers ou en stages de révisions et là les lacunes de certains sont "impressionnantes". Quelques-uns en première ont utilisé 1 fois Algobox en seconde… d'autres n'ont toujours aucune idée de ce qu'est une variable ou une boucle en terminale. Et d'autres ont des énoncés, des sujets de DM ou des corrigés qui sont faux ou avec des algorithmes qui ne fonctionnent pas. Ayant enseigné les mathématiques mais étant aussi "intéressé" par la programmation (créations de plusieurs sites web, programmes et mini jeux) je vais essayer dans ce document de repartir sur "les bases des bases" de l'algorithmique et ce qui fait, en tout cas pour ma part, que les élèves comprennent le fonctionnement des algorithmes et leur utilité.

II. Définition / But

Dans notre cas un algorithme aura pour but d'automatiser des taches ou de répondre à un problème mathématique. Il faut comprendre qu'avant de se lancer dans la rédaction d'un algorithme on doit d'abord avoir le raisonnement mathématique ou logique nécessaire pour résoudre le problème ou automatiser une tache. En effet les élèves placés face à un algorithme ont une fâcheuse tendance à le remplir au "pifomètre". S'il y a une expression de fonction dans l'énoncé la plupart essaieront de la replacer quelque part sans même comprendre pourquoi et à quoi sert l'algorithme. Dans le cas d'un algorithme à écrire complètement une majorité est perdue car les élèves vont d'abord chercher les variables nécessaires, alors qu'en raisonnant dans un premier temps de façon "humaine" il sera ensuite beaucoup plus simple de faire la transcription vers l'algorithme.

III. Notation / Légende

Dans le document les explications sont écrites en Times New Roman (comme cette phrase). Les algorithmes sont écrits en Arial (comme cette phrase). J'ai privilégié le pseudo-code rencontré le plus souvent dans les manuels et sujets de bac au lycée bien qu'il n'y ait rien "d'officiel".

IV. Le pseudo-code

Dans la « vraie vie » les algorithmes sont codés dans différents langages (Java, C++, Python, PHP, …). Il serait bon de faire une introduction sur ce point avec les élèves pour montrer la « puissance » de l'algorithmique et la place prépondérante qu'ils occupent dans notre vie quotidienne (smartphones, internet, …, absolument tous les élèves ont déjà utilisé un programme et donc vu une utilité de l'algorithmique, il faut leur faire "réaliser").

Afin de ne pas favoriser un langage et surtout de ne pas avoir à connaitre les usages propres à chaque langage on utilise ce qu'on appelle « un pseudo-code » qui est une façon de décrire un algorithme sans référence à un langage de programmation en particulier.

Ainsi je conseille dans un premier temps de laisser l'élève écrire les instructions comme il le souhaite pour les algorithmes puis de formaliser ensuite pour qu'il puisse travailler sur des exercices et que toute la classe utilise le même pseudo code.

Voici un exemple d'algorithme en pseudo-code à gauche et dans un langage réel (PHP) à droite :

Exemple pseudo-code	Exemple PHP (avec les variables provenant d'une autre page)

Variables : a,b,c réels
Début algorithme
Afficher("Entrer la longueur de AB")
Lire a
Afficher("Entrer la longueur de AC")
Lire b
Afficher("Entrer la longueur de BC")
Lire b
Si(a² + b² == c²) alors
Afficher("Le triangle ABC est rectangle en A")
Fin algorithme
	< ?php
	$a = $_POST["a"];
	$b = $_POST["b"];
	$c = $_POST["c"];
	if((pow($a,2)) + (pow($b,2) == (pow($c,2))
	{
	  echo "Le triangle ABC est rectangle en A";
	}
	?>

Les différentes parties seront détaillées par la suite. Les mots en gras correspondent aux différentes phases de l'algorithme et sont détaillés à la partie X.

V. Le texte

Un algorithme peut afficher du texte. En pseudo-code on utilisera l'instruction Afficher "Texte à afficher" le texte à afficher doit être mis entre guillemets pour que l'algorithme comprenne qu'on parle de texte et non pas d'une variable (voir paragraphe suivant).

VI. Les variables

Les variables vont servir dans un algorithme à "stocker des valeurs", que ce soit des nombres ou du texte. Elles sont stockées dans la mémoire de la machine qui fait fonctionner l'algorithme.

Dans une partie des langages de programmation et dans le pseudo-code que nous utiliserons, les variables doivent être "déclarées", c'est-à-dire qu'on doit prévenir l'algorithme que nous allons utiliser telle et telle variable. Ceci n'est pas forcément très intuitif pour les élèves et même pour certains enseignants.

Il faut faire comprendre aux élèves qu'ils peuvent commencer directement par l'algorithme mais que dès qu'ils auront besoin d'une variable il faudra l'ajouter au début de l'algorithme dans la partie "Variables".

De même dans certains langages et en pseudo-code le "type de variable" doit être déclaré. Au lycée on pourra généralement s'intéresser uniquement au type entier naturel, entier relatif et réel : soit la variable sera un entier soit elle n'en sera pas un et on parlera donc de réel.

Je pense qu'au moment d'aborder l'initialisation il est indispensable de faire un point avec les élèves sur le pourquoi de la déclaration d'un type (économiser les ressources, la mémoire de la machine qui fait fonctionner l'algorithme, …).

Résumé : on peut commencer et c'est plus simple par créer l'algorithme mais dès qu'on utilisera une variable on devra la déclarer avec son type (entier, réel, …) au début de celui-ci.

VII. Initialisation des variables

Ce n'est pas indispensable (suivant le langage utilisé) mais si on souhaite préciser la valeur de départ d'une variable on doit "l'initialiser". Si on a par exemple besoin d'une variable u pour un calcul sur une suite avec $u_0 = 4$, on notera :

Début algorithme

Variables :

u réel

Inititalisation :

u prend la valeur 4

Fin algorithme

VIII. Les entrées utilisateurs

Dans un algorithme on a généralement besoin de données apportées par celui qui l'utilise, c'est ce qu'on appelle une "entrée", on va "lire" la variable. Si on reprend l'exemple de la partie pseudo-code on voulait les longueurs des trois côtés d'un triangle (a, b et c) on note donc :

```
Début algorithme
Entrées :
Lire a
Lire b
Lire c
Fin algorithme
```

IX. Travail sur les variables

Ce qu'il faut ensuite comprendre c'est qu'une variable, comme son nom l'indique, ne va pas nécessairement garder la même valeur tout le long de l'algorithme. Elle sera modifiée avec différents calculs. Pour afficher une variable --> Afficher nom_variable avec nom_variable le nom de la variable. Il n'y a pas de guillemet.

Exemple 1 :

```
Début algorithme
Variables :
a réel
Traitement :
a prend la valeur 2
a prend la valeur (a + a)²
Fin algorithme
```

Dans cet exemple à la fin de l'algorithme a vaut 16 mais l'algorithme n'affiche rien.

Exemple 2 :

```
Début algorithme
Variables :
a réel
Traitement :
a prend la valeur 2
a prend la valeur (a + a)²
Sortie :
Afficher a
Fin algorithme
```

Cet exemple est quasiment identique au précédent mais dans ce cas l'algorithme affiche 16.

X. Les différentes parties de l'algorithme

Comme on peut le remarquer dans les algorithmes précédents et bien que ceci ne soit pas toujours fait dans le pseudo-code (et très rarement dans les langages de programmation les plus classiques), on peut indiquer les différentes "parties" de l'algorithme (toutes les parties ne sont pas nécessairement présentes) :

Début algorithme : on indique que l'algorithme débute…

Variables : on donne à l'algorithme le nom et le type des variables qui seront utilisées.

Initialisation : on indique les valeurs initiales des variables (facultatif et parfois incluse dans Traitement).

Entrées : on récupère des données entrées par l'utilisateur.

Traitement : l'algorithme effectue les calculs, boucles, vérifications, etc.

Sortie(s) : correspond à ce qui va être affiché au final par l'algorithme

Fin algorithme : on indique que l'algorithme est terminé.

Toutes les parties ne sont pas notée suivant les manuels / sujets. On retrouve le plus souvent les parties **Variables** et **Traitement**.

XI. Les boucles

Les boucles sont un élément indispensable des algorithmes. Elles permettent de répéter plusieurs fois la même opération. Pour le programme du lycée ces boucles pourront par exemple nous intéresser pour les calculs sur des suites et de façon générale pour tout ce qui nécessitera une répétition.

1. La boucle "Pour"

La boucle "Pour" répète une partie de l'algorithme pour une valeur d'une variable à une autre. Structure :

```
Pour nom_variable allant de o à p (avec un pas de q)
Instructions
Fin Pour
```

- nom_variable le nom de la variable qui sera utilisé pour la boucle (souvent noté i ou j mais qui peut être une variable autre).
- o la première valeur de la variable, ainsi nom_variable prendra cette valeur pour la première boucle. Si la variable est i et que o est égal à 3 alors les calculs de la première boucle se feront avec i = 3.
- q le pas, à la fin de chaque boucle nom_variable augmentera de q, la partie (avec un pas de q) est notée entre parenthèses car elle n'est pas obligatoire. En effet même en ne précisant rien la boucle "Pour" fait augmenter nom_variable de 1 à la fin de chaque boucle. Toutefois si on souhaite un pas différent de 1 il faut le préciser.
- p la dernière valeur de la variable, quand nom_variable atteint cette valeur on entre une dernière fois dans la boucle, puis on "continue" l'algorithme.

Exemple :

Début algorithme

Variables :

u réel

i entier

Traitement :

u prend la valeur 1

i prend la valeur 1

Pour i allant de 1 à 3 avec un pas de 1

u prend la valeur u x i²

Fin Pour

Sortie :

Afficher u

Fin algorithme

Essayons de comprendre l'algorithme --> quand il y a des boucles, il faut "raisonner" comme un ordinateur :

Position par rapport à la boucle pour :	u	i	Affichage
Avant la boucle 1	u = 1	i = 1	Néant
(On entre forcément dans la première boucle avec i = 1) A la fin de la boucle 1	u = 1 x 1² = 1	i = 2	Néant
(On se demande si on entre dans la boucle : i = 2, on est bien entre 1 et 3 donc oui) A la fin de la boucle 2	u = 1 x 2² = 4	i = 3	Néant
(On se demande si on entre dans la boucle : i = 3, on est bien entre 1 et 3 donc oui) A la fin de la boucle 3	u = 4 x 3² = 36	i = 4	Néant
(i = 4, on n'entre plus dans la boucle pour, on peut regarder la suite de l'algorithme)	36	4	36

On notera que pour la boucle "Pour", le pas (appelé également l'incrément) n'est généralement pas indiqué. Dans ce cas cela signifie que la variable utilisée augmente de 1 à chaque boucle.

Exemple :

Début algorithme

Variables :

u réel

k entier

Traitement :

u prend la valeur 1

k prend la valeur 3

Pour k = 3 à 5

u prend la valeur u x k

Afficher u

Fin pour

Fin algorithme

Position par rapport à la boucle pour :	u	k	Affichage
Avant la boucle 1	1	3	Néant
(On entre forcément dans la première boucle avec k = 3) A la fin de la boucle 1	u = 1 x 3 = 3	4	3
(On se demande si on entre dans la boucle : k = 3, on est bien entre 3 et 5 donc oui) A la fin de la boucle 2	u = 3 x 4 = 12	5	12
(On se demande si on entre dans la boucle : k = 4, on est bien entre 3 et 5 donc oui) A la fin de la boucle 3	u = 12 x 5 = 60	6	60
(k = 5, on n'entre plus dans la boucle pour, on peut regarder la suite de l'algorithme)	60	6	Néant

Si dans l'exemple précédent, on met le "Afficher u" après la boucle alors l'algorithme n'affichera que la dernière valeur c'est-à-dire 60.

2. La boucle "Tant que"

Comme son nom l'indique cette boucle va effectuer une instruction tant qu'une condition est vérifiée.

Exemple :

Début algorithme

Variables :

u réel

n entier

Traitement :

u prend la valeur 0,5

n prend la valeur 0

Tant que u > 0,06 faire

u prend la valeur u^2

n prend la valeur n+1

Fin tant que

Sortie :

Afficher n

Fin algorithme

Position par rapport à la boucle Tant que :	u	n	Affichage
Avant le 1er Tant que	0,5	0	Néant
(u = 0,5 donc on est bien dans le cas u > 0,06, on rentre dans le 1er Tant que) A la fin du Tant que n°1	$u = 0,5^2 = 0,25$	1	Néant
(u = 0,25 donc on est bien dans le cas u > 0,06, on rentre dans le 2ème Tant que) A la fin du Tant que n°2	u = 0,25 x 0,25 = 0,0625	2	Néant
(u = 0,0625 donc on est bien dans le cas u > 0,06, on rentre dans le 3ème Tant que) A la fin du Tant que n°3	u = 0,0625 x 0,0625 = 0,00390625	3	Néant
(u = 0,00390625 < 0,06, on n'entre plus dans le Tant que, on peut regarder la suite de l'algorithme)	0,00390625	3	3

Cet algorithme permet d'afficher par exemple le rang à partir duquel le terme de la suite $u_{n+1} = u_n^2$ est inférieur à 0,06.

XII. Les conditions

1. Les conditions simples

Dans les algorithmes niveau lycée, les conditions sont généralement traités uniquement avec l'instruction "Si Alors Sinon". L'instruction est réalisée si une condition est vérifiée.

Exemple :

```
Début algorithme
Variables :
x réel
Entrées :
Lire x
Traitement :
Si (x < 0) alors                    *cf. opérateurs de comparaison ci-dessous
x prend la valeur –x
Fin si
Sortie :
Afficher x
Fin algorithme
```

Dans l'exemple si la valeur entrée par l'utilisateur est négative alors l'algorithme fait prendre la valeur –x à la variable. L'algorithme permet de renvoyer la valeur absolue de la variable entrée par l'utilisateur.

La condition à vérifier est testée grâce à des opérateurs de comparaison :

==	Vérifie une égalité
<>	Vérifie que deux variables sont différentes
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à

Enseigner l'algorithmique au lycée. Les bases et méthodes. Les difficultés rencontrées et les solutions. blog.partiprof.fr - Jérémy Tellier

2. Les conditions multiples

On peut dans l'instruction "Si" ajouter plusieurs conditions avec ce qu'on nomme "des opérateurs logiques" : OU et ET. On peut par exemple vérifier qu'une variable appartient à un intervalle :

```
Début algorithme
Variables :
x réel
Entrées :
Lire x
Traitement :
Si (x > 0 ET x < 10)  alors
Afficher "x appartient à ]0 ; 10["
Sinon
Afficher "x n'appartient pas à l'intervalle ]0 ; 10["
Fin si
Sortie :
Afficher x
Fin algorithme
```

L'algorithme précédent demande un nombre à l'utilisateur et indique si celui-ci appartient à l'intervalle]0 ; 10[.

XIII. La géométrie

Bien que cela soit très variable suivant les langages de programmation nous considèrerons que les algorithmes peuvent placer des points et tracer des segments (ce qui est le cas avec Algobox et les calculatrices).

Exemple :

```
Début algorithme
Placer un point A en (0 ; 0)
Placer un point B en (2 ; 2)
Tracer le segment [AB]
Fin algorithme
```

L'algorithme place un point A aux coordonnées (0 ; 0), un point B en (2 ; 2) puis les relie par un segment.

XIV. Nombres aléatoires

L'intérêt des algorithmes est de pouvoir simuler le "hasard". On pourra ainsi écrire :

```
a prend la valeur d'un nombre entier aléatoire compris entre 1 et 6
```

Cette ligne simule un lancer de dé équilibré à 6 faces. Sur Algobox ALEA_ENT(p,n) permet d'obtenir un nombre entier aléatoire compris entre p et n.

XV. Quelques exemples

1. Niveau seconde

Exemple 1 :

Ecrire un algorithme permettant de retrouver le centre de ce cercle :

Indice : on rappelle que le centre du cercle circonscrit à un triangle est le point d'intersection de ses médiatrices.

Exemple 2 :

Compléter l'algorithme ci-dessous.

Début algorithme
Variables
x réel
Traitement
Si ……
Alors
Afficher "x appartient à]- ∞ ; 4]
Fin algorithme

Un livre pour la seconde :

Ou plutôt le "cahier" car pour le moment je n'en ai vu qu'un qui soit acceptable au niveau du prix et du travail proposé pour des élèves de seconde (bien que les bases de l'algorithmique ne soient pas suffisamment posées à mon goût) : le cahier d'algorithmique de Bordas qui offre des niveaux très variés et beaucoup d'algorithmes à compléter.

Enseigner l'algorithmique au lycée. Les bases et méthodes. Les difficultés rencontrées et les solutions. blog.partiprof.fr - Jérémy Tellier

2. Niveau première S

Exemple 1 :

On souhaite obtenir le signe d'une fonction polynôme de degré 2 de la forme $ax^2 + bx + c$. L'utilisateur entrera les valeurs de a, b et c. Puis l'algorithme devra afficher le signe de la fonction suivant les valeurs de x.

On notera que cet algorithme est relativement long et complexe à mettre en place pour des élèves de première S. Il pourra être plus ou moins guidé.

Exemple 2 :

Ecrire un algorithme où :

- L'ordinateur choisit un nombre entier au hasard compris entre 1 et 1000.
- L'utilisateur doit trouver le nombre. S'il entre un nombre trop petit, l'algorithme lui affiche « Plus grand », s'il entre un nombre trop grand l'algorithme lui affiche « Plus petit », ceci jusqu'à ce qu'il trouve le nombre. Si le nombre entré par l'utilisateur est le même que celui choisi au hasard l'algorithme affiche « Gagné ».

Puis en dernier lieu, modifier l'algorithme pour qu'il affiche en combien de coup(s) le nombre a été trouvé.

3. Niveau terminale S

Exemple 1 :

On considère l'algorithme :

```
A et C sont des entiers naturels,
C prend la valeur 0
Répéter 9 fois
        A prend une valeur aléatoire entière entre 1 et 7.
                Si A > 5 alors C prend la valeur de C + 1
                Fin Si
Fin répéter
Afficher C.
```

Dans l'expérience aléatoire simulée par l'algorithme précédent, on appelle X la variable aléatoire prenant la valeur C affichée.
Quelle loi suit la variable X? Préciser ses paramètres.

Enseigner l'algorithmique au lycée. Les bases et méthodes. Les difficultés rencontrées et les solutions. blog.partiprof.fr - Jérémy Tellier

Exemple 2 :

La suite $(u_n)_{n\geq 1}$ est définie par :

$$u_n = \frac{2n+1}{3n-1}$$

Ecrire un algorithme qui permet de calculer le terme de rang n.

Enseigner l'algo... Les bases et m... ...cultés rencontrées et ...tiprof.fr - Jérémy Te...

Exemple 3 :

Entrée	Saisir un réel strictement positif non nul a Saisir un réel strictement positif non nul $b (b > a)$ Saisir un entier naturel non nul N
Initialisation	Affecter à u la valeur a Affecter à v la valeur b Affecter à n la valeur 0
Traitement	TANT QUE $n < N$ Affecter à n la valeur $n+1$ Affecter à u la valeur $\frac{a+b}{2}$ Affecter à v la valeur $\sqrt{\frac{a^2+b^2}{2}}$ Affecter à a la valeur u Affecter à b la valeur v
Sortie	Afficher u, afficher v

Reproduire et compléter le tableau suivant, en faisant fonctionner cet algorithme pour $a = 4, b = 9 et N = 2$. Les valeurs successives de u et v seront arrondies au millième.

n	a	b	u	v
0	4	9		
1				
2				

XVI. Problèmes courants

1. Noms de variables

J'ai abordé dans la partie VII les suites. Ces suites posent couramment problème aux élèves dans les algorithmes car ils vont par exemple vouloir noter une variable u_n. Malheureusement bien qu'on le trouve dans certains pseudocodes (tout comme y_A ou x_A pour des coordonnées même dans des sujets d'examens…) aucun langage ne permettra de mettre une partie du nom de la variable en indice.

Il faut faire comprendre aux élèves que les variables ne peuvent pas être notées avec une partie en indice. Pour reprendre les exemples précédents on peut noter uniquement u, x et y ou si on souhaite conserver plus d'informations on peut noter u_n, x_a et y_a.

2. Les boucles

Les boucles sont très peu intuitives pour la plupart des élèves. Aussi dans un premier temps on peut faire des exemples avec des boucles "courtes" où les élèves doivent inscrire le numéro de la boucle et l'état des variables à chaque instant. Ceci permet une meilleure compréhension.

3. Les "meilleurs"

Bien que les algorithmes rebutent la plupart des élèves et les mettent assez facilement en difficulté, pour certains élèves ceux-ci paraissent très voire trop simples. On peut d'ailleurs noter qu'en réalité le niveau demandé en algorithmique est faible voire très faible pour les élèves de terminale S par rapport au reste du programme. On peut espérer qu'il augmente en même temps que le niveau des élèves durant les prochaines années.

Pour ces élèves il faut bien évidemment prévoir des choses plus compliquées. Heureusement ce n'est pas ce qui manque. Quitte à leur demander de les programmer sur leurs calculatrices ou ordinateurs et en leur donnant uniquement le sujet et/ou une problématique.

XVII. Conclusions

Il est regrettable que l'Education Nationale ait fait le choix de l'enseignement de l'algorithmique de façon "bâclée". Tous les professeurs de mathématiques ne sont pas aptes à l'enseigner. Au vu de la place croissante que prend l'algorithmique, la programmation et l'informatique de façon générale, ces matières devraient être enseignées "à part", par des professeurs dédiés et formés pour cela.

Concernant l'algorithmique on notera qu'Algobox est l'un des programmes les plus utilisés et les plus "plébiscités" au lycée mais il en existe des tas d'autres. On pourra citer Scratch. Et bien que le temps manque souvent pour terminer le programme, l'algorithmique ne peut être enseignée sans réaliser de "vrais programmes" que ce soit sur Algobox ou en PHP par exemple qui demande peu de "choses" à installer et peuvent se révéler assez puissants. Certains lycées ont des projets de club d'informatique ou de programmation. Dans ce cas c'est l'idéal les projets peuvent être beaucoup plus poussés (mini-jeux, site web dynamique avec comptes utilisateurs, …).

Nous pourrons également noter que ce document se veut succinct et ne rentre pas dans les détails de tous les algorithmes et "éléments" qui pourraient être utiles. Les évolutions de l'enseignement de l'algorithmique devront dans tous les cas être nombreuses dans les années à venir afin d'offrir un enseignement de qualité.

Enseigner l'algorithmique au lycée.
Les bases et méthodes.
Les difficultés rencontrées et les solutions.
blog.partiprof.fr - Jérémy Tellier