Mise en place d'un outil pédagogique pour l'enseignement de l'algorithmique

par Jean-Alain RODDIER
IA-IPR de mathématiques
Académie de Clermont-Ferrand

C'est dans le cadre d'un enseignement en construction au niveau lycée que nous proposons de conduire une réflexion destinée à la mise en place d'outils pédagogiques propres à faciliter la présentation d'algorithmes.

Ce document est réalisé dans le cadre d'un séminaire dédié à l'enseignement de la spécialité Informatique et Sciences du Numérique qui se déroule à Nancy le 11 et 12 avril 2013. Il se veut être une base de travail et à ce titre, il a vocation à être enrichi dans le temps afin qu'il puisse progressivement servir de référence dans le développement au quotidien de « bonnes » pratiques pour l'enseignement de la spécialité ISN.

Nous choisissons de présenter une méthode utile à la présentation d'un algorithme en classe avec les élèves. Cette méthode a vocation à être mise en place en « co-construction » avec les élèves et elle consiste à suivre des scénarii basés sur les six phases didactiques suivantes :

- la découverte de l'algorithme où il s'agit de créer de l'appétence à la situation ;
- l'appropriation de l'algorithme sur un exemple ;
- sa transcription en langage naturel ;
- son écriture en langage pseudocode ;
- son développement en langage de programmation ;
- l'observation de l'affectation des variables tout au long du déroulement du pmù

rogramme.

La description générale

La méthode présentée ici repose sur le principe que toute présentation d'un point nouveau est effectuée en parallèle d'un point d'appui, ainsi :

- la transcription de l'algorithme est réalisée en appui d'une appropriation réelle du traitement d'un exemple ;
- le développement en langage pseudocode puis de programmation est construit en appui de la transcription en langage naturel.

1) La découverte de l'algorithme : créer une situation d'appétence à la situation

Il s'agit de réserver en préambule de la présentation d'un algorithme une partie visant à mettre en place l'enjeu de cet algorithme dans un cadre général. Cette partie pourra être construite autour d'une problématique rencontrée à l'intérieur d'un projet. Il s'agit de mettre en avant des besoins puis d'introduire (ou de faire émerger) la solution algorithmique sans rentrer – pour l'instant - dans les détails.

2) L'appropriation de l'algorithme. : mettre en avant le traitement d'un exemple

On prend ici le contre-pied de ce qu'est habituellement un exemple en mathématiques où l'exemple vient la plupart du temps illustrer une propriété globale. Dans cette phase, on souhaite faire « tourner » l'algorithme étudié sur un exemple et décortiquer ce traitement en préambule de tout exposé à caractère général.

a) Traitement en continu et traitement pas à pas

Dans un premier temps, le traitement de l'exemple peut être conduit dans sa globalité : on part de l'entrée et l'on aboutit à la sortie. Il est possible de le développer ensuite pas à pas afin d'en faire ressortir les tenants et les aboutissants ainsi que tous les éléments techniques.

b) Des questions sur cet exemple

On veille à mettre en place une structure autour du « Lire » et « Dire » afin d'expliciter l'exemple traité et de faire ressortir des éléments structurant de l'algorithme.

« *Que fait l'algorithme ?* » ;

« *Quel est l'élément à fournir au départ ?* » ; « *Quel est l'élément produit à l'arrivée ?* »

« *Y-a-t-il un processus répétitif ? Si oui, quel est-il ?* »

« *L'algorithme s'arrête-t-il ? Si oui, qu'est-ce qui fait qu'il s'arrête ?* »

c) La décortication « concrète » de l'exemple

Les différentes réponses apportées à la question posée peuvent conduire à une décortication effective au feutre de l'exemple et la mise en place de ce que nous appellerons (entre professeurs) un code couleurs.

On pourra ainsi :

Repasser en rouge l'élément de départ.

Repasser en bleu le processus répétitif.

Entourer en orange la condition d'arrêt.

Repasser en vert l'élément produit à l'arrivée.

d) Le traitement par l'élève d'un exemple personnel

L'élève est mis en activité à partir d'un exemple qu'il choisit. Une fois l'algorithme « déroulé » sur cet exemple, le professeur le conduit à utiliser les 4 couleurs mentionnées précédemment afin de faire ressortir les fondements de l'algorithme.

3) La traduction de l'algorithme en langage naturel

a) Sa légitimité

Les élèves peuvent penser que le fait d'avoir un exemple suffit à passer directement au développement informatique de l'algorithme. Il est utile de montrer que l'écriture en langage naturel est une étape centrale qui permet :

- de rendre compte de l'algorithme sans parler d'informatique ;
- de conduire une phase de transition entre l'exemple et la programmation ;

Nous citons à dessein la phrase suivante d'Albert Camus « *Un exemple n'est pas forcément un exemple à suivre* » afin de souligner que l'écriture de l'algorithme en langage naturel permet de conduire un raisonnement déductif afin de « contrôler » (pour ne pas dire démontrer) sa véracité dans le cas général.

b) L'écriture en langage naturel

Nous proposons de réaliser cette écriture en mettant en correspondance l'exemple « décortiqué » et l'algorithme en langage naturel, pour cela on peut présenter les choses sous la forme d'un tableau à deux colonnes avec à droite une colonne vide (au départ) et de l'autre côté l'exemple réécrit en utilisant le code couleurs. Ce dernier vient signaler les correspondances entre la colonne de droite et celle de gauche.

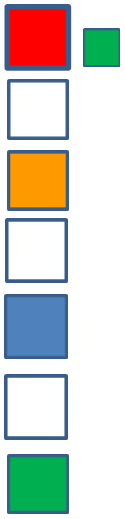
Langage naturel	**Exemple**
	

Soulignons un point important à montrer aux élèves : contrairement à ce qu'il se passe sur l'exemple où la condition d'arrêt vient à la fin du traitement, elle doit être – la plupart du temps - placée dans l'écriture en langage naturel avant la description du processus répétitif car elle signale la condition d'arrêt de celui-ci.

Pour certains algorithmes, il est nécessaire d'évoquer le résultat en début de déclaration en langage naturel, c'est le cas en particulier lorsque l'on a une liste à prévoir qui sera vide au départ et remplie progressivement.

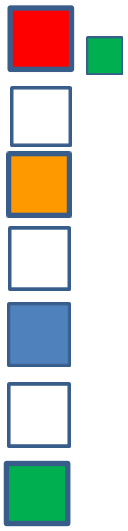
4) L'écriture de l'algorithme en langage pseudocode

L'idée est de suivre le même mode de présentation pour écrire l'algorithme en pseudo code avec au départ un tableau à deux colonnes une pour le programme, une autre pour l'écriture en langage naturel. L'utilisation des codes couleurs pour souligner les correspondances entre tel élément de la colonne de droite et son pendant dans la colonne de gauche.

Langage pseudocode	Langage naturel
	

5) Le développement en langage de programmation

En respectant la même présentation sous la forme d'un tableau à deux colonnes, on développe le programme dans la colonne de gauche en appui de l'écriture de l'algorithme en langage pseudocode. Des éléments d'explications de la syntaxe du langage de programmation pourront être apportés dans la colonne de droite de façon à rendre davantage explicite cette phase de programmation.

Langage de programmation	Langage pseudocode
	

6) L'observation des valeurs des variables

Après la phase de programmation, il est utile de mettre en place une phase de déroulement pas à pas du programme qui permet de voir (et de contrôler) les différentes affectations de toutes les variables considérées. On pourra développer cette partie sur l'exemple pris au tout début puis sur l'exemple personnel considéré par l'élève. La présentation de cette observation est faite sous la forme du tableau ci-dessous qui reprend les 4 codes couleurs.

Par souci d'efficacité dans la lecture de ce tableau, nous avons pris – à titre indicatif - des exemples de noms de variables (N, i , k , S , L).

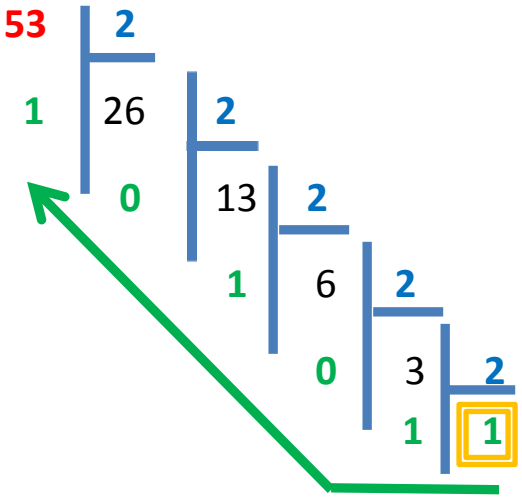
Etape	N	i	k	S	L
0 (initialisation)		1	0	0	[]
1	(à remplir)	2	(à remplir)	(à remplir)	(à remplir)
2	"	"	"	"	"
(à remplir)	"	"	"	"	"
1000 (arrêt)	"	1001	"	"	

Une mise en œuvre de la méthode sur un premier exemple : le codage binaire

1) La découverte de l'algorithme

Celle-ci peut se faire par exemple au moyen de considérations sur le traitement de l'image où l'on entrevoit dans la pratique le fait de ramener toutes les informations suffisantes à la transmission d'une image sous la forme d'un code en binaire.

2) Le traitement d'un exemple : 53



53 divisé par 2 donne 26 reste 1

26 divisé par 2 donne 13 reste 0

13 divisé par 2 donne 6 reste 1

6 divisé par 2 donne 3 reste 0

3 divisé par 2 donne 1 reste 1

3) L'écriture en langage naturel

Langage naturel	Exemple
On prend un entier N>1 On le divise successivement par 2 Jusqu'à obtenir un quotient égal à 1 Le résultat est obtenu en mettant bout à bout 1 suivi des restes dans l'ordre inverse où on les a obtenus.	53 53 divisé par 2 donne 26 reste 1 26 divisé par 2 donne 13 reste 0 13 divisé par 2 donne 6 reste 1 6 divisé par 2 donne 3 reste 0 3 divisé par 2 donne 1 reste 1 1 1 0 1 0 1

Les cas N=0 et N=1 sont traités au moment où les élèves sont amenés à prendre leur propre exemple. On intègre ces cas particuliers à l'écriture de l'algorithme en langage naturel.

4) L'écriture en pseudocode

Pseudocode	Langage naturel
On définit une fonction de N Si N=0, renvoyer le résultat 0 Si N=1, renvoyer le résultat 1 q ← E(N/2) et r ← Mod(N,2) listerésultat ← {r} Répéter tant que q≠1 N ← E(N/2) q ← E(N/2) et r ← Mod(N/2) listerésultat ← listerésultat + {r} fin de tant que listerésultat ← listerésultat + {1} listerésultat ← listerésultat renversée Renvoyer la listerésultat	On prend un entier N Si N=0, renvoyer le résultat 0 Si N=1, renvoyer le résultat 1 On commence par diviser N par 2 On prépare la liste résultat Jusqu'à obtenir un quotient égal à 1 On divise successivement par 2 On prépare la liste résultat Le résultat est obtenu en mettant bout à bout 1 suivi des restes dans l'ordre inverse où on les a obtenus.

5) Le développement en langage Python

Langage Python	Pseudocode
>>> def bin(N) : assert(N==int(N)) if N==0 : return 0 if N==1 : return 1 q = int(N/2) r = N%2 listeresultat = [r] while q !=1 : N = q q = int(N/2) r = N%2 listeresultat = listeresultat+[r] listeresultat = listeresultat+[1] listeresultat = listeresultat.reverse() return listeresultat	On définit une fonction de N # On teste le caractère entier de N Si N=0, renvoyer le résultat 0 Si N=1, renvoyer le résultat 1 q ← E(N/2) r ← Mod(N,2) listerésultat ← {r} Répéter tant que q≠1 N ← E(N/2) q ← E(N/2) r ← Mod(N/2) listerésultat ← listerésultat + {r} fin de tant que listerésultat ← listerésultat + {1} listerésultat ← listerésultat renversée Renvoyer la liste listerésultat

6) La gestion des variables

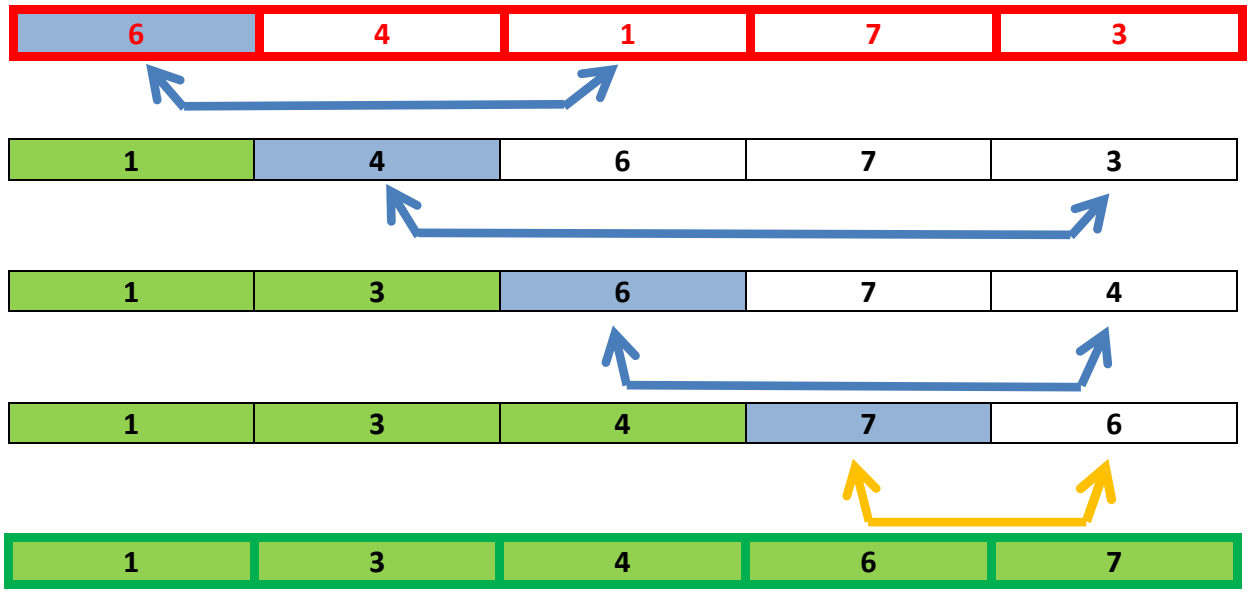
N° de l'étape	N	q	r	listeresultat
0	53	26	1	[1]
1	26	13	0	[0 , 1]
2	13	6	1	[1 , 0 ,1]
3	6	3	0	[0 , 1 , 0 , 1]
4	3	1	1	[1, 0 , 1 , 0 , 1]
5				[1,1,0,1,0,1]
6				[1,0,1,0,1,1]

Une mise en œuvre de la méthode sur un deuxième exemple : le tri par sélection

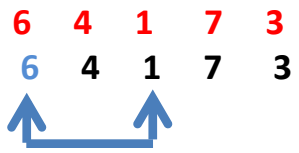
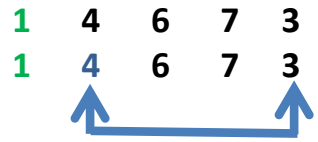
1) La découverte de l'algorithme

Il est utile de légitimer l'importance de l'utilisation d'un mode pensé de tri par souci d'efficacité.

2) Traitement d'un exemple :



3) L'écriture de l'algorithme en langage naturel

Langage naturel	Exemple
On part d'une liste On sélectionne le premier élément de la liste et on l'inverse (si besoin est) avec l'élément le plus petit du reste de la liste.	
On sélectionne le deuxième élément de la liste et on l'inverse (si besoin est) avec l'élément le plus petit du reste de la liste.	
On poursuit le même procédé	
	
jusqu'à sélectionner l'avant dernier élément de la liste et l'inverser si besoin est avec le dernier élément de la liste.	
On obtient la liste triée dans l'ordre croissant.	

4) L'écriture en langage pseudocode

Pseudocode	Langage naturel
On définit une fonction de T **n ← longueur de la liste T** Pour i variant de 0 jusqu'à **n-2** imin ← i Tmin ← T[i] Pour j variant de i+1 à n-1 Si T[j] < Tmin alors imin ← j Tmin ← T[j] Prendre le j suivant T[i], T[imin] ← Tmin, T[i] Prendre le i suivant Renvoyer la liste T	On part d'une liste *(on la note T)* *On note n la longueur de la liste* On sélectionne le premier élément de la liste *(c'est T[0])* et on l'inverse (si besoin est) avec l'élément le plus petit *(Tmin)* du reste de la liste. On sélectionne le deuxième élément de la liste *(c'est T[1])* et on l'inverse (si besoin est) avec l'élément le plus petit *(Tmin)* du reste de la liste. On poursuit le même procédé jusqu'à sélectionner l'avant dernier élément de la liste *(c'est T[n-2])* et l'inverser si besoin est avec le dernier élément de la liste. On obtient la liste triée dans l'ordre croissant.

5) Le développement en langage Python

Langage Python	Pseudocode

>>> def tri(T) :
 	
n=len(T)
 	
for i in range (0, n-1) :
 	
# sur python, i varie de 0 à n-2 (compris) 	
imin = i
Tmin = T[i]
For j in range(i+1, n) :
# j varie de i+1 à n -1 	
If T[j]<Tmin :
Imin =j
Tmin = T[j]
 	
T[i],T[imin] = Tmin,T[i]
 	
return(T)
	Lire la liste T (entrer cette liste sous la forme d'un n-uplet)
	n ← longueur de la liste T
	
	Pour i variant de 0 jusqu'à **n-2**
	
	imin ← i
	Tmin ← T[i]
	Pour j variant de i+1 à n-1
	
	Si T[j] < Tmin alors
	imin ← j
	Tmin ← T[j]
	Prendre le j suivant
	T[i], T[imin] ← Tmin, T[i]
	Prendre le i suivant
	Renvoyer la liste T

6) La gestion des variables

N° étape	T	i	imin	Tmin	j	T[j]
0	6 4 1 7 3	0	0	6	1	4
1	6 4 1 7 3	0	1	4	2	1
2	6 4 1 7 3	0	2	1	3	7
3	6 4 1 7 3	0	2	1	4	3
4	1 4 6 7 3	1	1	4	2	6
5	1 4 6 7 3	1	1	4	3	7
6	1 4 6 7 3	1	1	4	4	3
7	1 3 6 7 4	2	2	6	3	7
8	1 3 6 7 4	2	2	6	4	4
9	1 3 4 7 6	3	3	7	4	6
10	1 3 4 6 7	(4)				

Conclusion

Au-delà d'un simple artifice de présentation d'un algorithme, il s'agit par la mise en œuvre de cette méthode d'inscrire la présentation des algorithmes dans une véritable dynamique explicative où les implicites sont levés par mise en regard de tel ou tel point nouveau avec un point acquis.

Souhaitons que le caractère évolutif de ce texte dans le temps lui permette dans sa mise en œuvre au quotidien d'accroître l'efficacité de l'enseignement de l'algorithmique.