COLLECTION ODYSSÉE

Cahier d'Activités 2de
ALGORITHMIQUE

Sous la direction de

Éric SIGWARD

IA-IPR de mathématiques de l'académie de Strasbourg

Auteurs

François BRISOUX

Professeur de mathématiques au lycée français René Descartes de Rabat

Christian BRUCKER

Professeur de mathématiques au lycée Théodore Deck de Guebwiller

Yvan MONKA

Professeur de mathématiques au collège Albert Camus de Soufflenheim

Éric SIGWARD

PROGRAMME DE MATHÉMATIQUES 2009

Édition réservée à l'enseignant.
Elle comporte des corrigés en rose qui ne figurent pas dans la version élève.

Hatier

AVANT-PROPOS

« Exercez-vous pour l'amour du ciel
à exécuter de petites choses. »
Épictète (50-130)

Les élèves ont rencontré au cours du collège un certain nombre d'algorithmes : les algorithmes opératoires comme la division euclidienne, le calcul avec les fractions, l'algorithme d'Euclide, des algorithmes d'enchaînement d'opérations mais également des algorithmes de constructions géométriques. Les nouveaux programmes des classes de lycée privilégient les activités qui mettent en œuvre des démarches algorithmiques dans les différentes notions étudiées : fonctions, géométrie, statistiques et probabilités.

Les exercices de ce cahier sont classés selon les trois grandes parties du programme de seconde : fonctions, géométrie et statistiques. Ils peuvent cependant être traités de façon indépendante. Les prérequis mathématiques, volontairement restreints, sont précisés en en-tête de chaque exercice.

Les compétences algorithmiques* mises en jeu et travaillées sont indiquées au début de chaque exercice. La plupart des algorithmes de ce cahier se traitent facilement sur une calculatrice programmable. La présence dans les énoncés de langages plus évolués comme le langage Python ou bien spécifiques à un logiciel comme Scilab ou Xcas, permet aux élèves de s'entraîner à la lecture d'algorithme car leur syntaxe est particulièrement simple et intuitive.

Le langage de programmation n'est pas la priorité de ces exercices, il est par contre essentiel de bien comprendre les démarches algorithmiques, de s'entraîner à élaborer des algorithmes en langage naturel, de comprendre le fonctionnement des algorithmes donnés dans le but de les modifier ou de les compléter dans le cas d'une généralisation du problème étudié.

Les auteurs.

* Ces compétences se réfèrent aux compétences identifiées dans le document *Ressources pour la classe de seconde générale et technologique – Algorithmique*, page 3.
▶ eduscol.education.fr/cid45766/mathematiques-pour-le-college-et-le-lycee.html

© HATIER, PARIS 2013 - ISBN : 978-2-218-96181-6
ISBN 978-2-218-96182-3
Toute représentation, traduction, adaptation ou reproduction, même partielle, par tous procédés, en tous pays, faite sans autorisation préalable est illicite et exposerait le contrevenant à des poursuites judiciaires. Réf.-: loi du 11 mars 1957, alinéas 2 et 3 de l'article 41. Une représentation ou reproduction sans autorisation de l'éditeur ou du Centre Français d'exploitation du droit de Copie (20, rue des Grands Augustins, 75006 Paris) constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

Initiation à l'algorithmique

1 Qu'est-ce qu'un algorithme ?

Un algorithme est une liste finie de processus élémentaires, appelés instructions élémentaires, amenant à la résolution d'un problème.

APPLICATION 1

Yasmine visite Paris mais elle n'a pas pris de plan. Elle se promène dans l'avenue de l'Observatoire, en direction du jardin du Luxembourg lorsqu'elle demande son chemin à un passant qui lui indique :

« Continuez jusqu'au bout, tournez à droite, prenez la première à gauche puis la troisième à droite, continuez alors tout droit jusqu'à la bifurcation de la rue, vous y êtes. »

a. Où souhaite se rendre Yasmine ?

Yasmine souhaite se rendre au Panthéon.

b. Une fois sa visite terminée, elle demande à nouveau son chemin à une fleuriste qui lui indique :

« Repartez dans la direction du jardin du Luxembourg et prenez la quatrième à droite. Continuez tout droit, franchissez la Seine puis tournez à la première à droite, avancez et vous y êtes. »

Quelle est la nouvelle étape de Yasmine ?

Notre-Dame de Paris.

c. Indiquer de la même façon à Yasmine le chemin à suivre pour retourner à son point de départ.

Faire demi-tour et longer la Seine, prendre le deuxième pont à gauche, aller tout droit jusqu'au bout du parc du Luxembourg, prendre à droite puis la deuxième à gauche.

Voici quelques algorithmes déjà rencontrés au cours de votre scolarité.

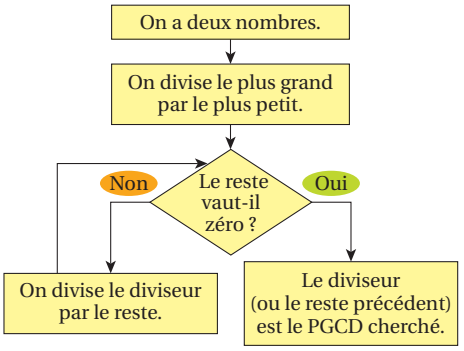
• L'algorithme de la division euclidienne

C'est une suite finie d'instructions qui calculent le quotient et le reste de la division de deux entiers. On répète une succession d'opérations tant que le quotient partiel est supérieur ou égal au diviseur.

• L'algorithme d'Euclide

C'est une suite finie de divisions euclidiennes aboutissant au calcul du PGCD de deux entiers.

En effet, pour déterminer le PGCD de deux entiers, on applique l'algorithme ci-contre.



EXEMPLE

Calculons le PGCD de 946 et 444 :

946 | 444	444 | 58	58 | 38	38 | 20	20 | 18	18 | 2
58 | 2	38 | 7	20 | 1	18 | 1	2 | 1	0 | 9

Le PGCD de 946 et 444 est donc 2.

REMARQUE

Un algorithme ne doit contenir que des instructions qui ne comportent aucune ambiguïté et donc compréhensibles par celui qui doit les exécuter.

2 Les langages

Un algorithme peut être décrit en langage « naturel », mais on utilise dans la plupart des cas un langage plus précis adapté aux instructions utilisées : on parle alors de langage de programmation.

Les exemples d'algorithmes de ce manuel sont écrits dans plusieurs environnements : le langage naturel, AlgoBox, le langage de programmation Python, le langage des calculatrices les plus courantes (TI et Casio), le logo avec le logiciel GéoTortue ainsi que les syntaxes des logiciels Xcas et Scilab.

On pourra se reporter au tableau de correspondance entre les différentes syntaxes utilisées (▶ **voir à la fin du cahier**).

De façon générale, on peut considérer trois étapes dans un algorithme :

1. **L'entrée des données**
Dans cette étape figure la lecture des données qui seront traitées au cours de l'algorithme. Ces données peuvent être saisies au clavier ou bien être lues dans un fichier annexe.

2. **Le traitement des données**
C'est le cœur du programme. Il est constitué d'une suite d'instructions, parmi lesquelles les différentes opérations sur les données, qui permettent de résoudre le problème.

3. **La sortie des résultats**
C'est le résultat obtenu qui peut être affiché à l'écran ou enregistré dans un fichier.

3 L'affectation

Une des instructions fondamentales est l'**affectation** d'une valeur à une variable.

L'affectation consiste à attribuer une valeur à une variable.

Les valeurs prises par les variables sont, par exemples, des nombres entiers, des nombres décimaux, des chaînes de caractères, des listes, des tableaux, des graphiques, etc., et elles sont susceptibles de changer au cours de l'algorithme.

L'affectation se traduit de différentes manières selon le langage. Par exemple, si une variable *A* doit être affectée de la valeur 3, on écrit en langage naturel : « *A* **prend** la valeur 3. »

Ce que l'on écrit :

3→A	A = 3	A:=3
en langage TI ou Casio ;	en langage Python ou sous Scilab ;	sous Xcas.

EXEMPLE

On considère la suite des affectations ci-contre :

```
a=1
b=5
a=b-3
b=2*a
a=b
b=a
```

Pour déterminer les valeurs prises par *a* et *b* à la suite de ces affectations, on exécute pas à pas la succession des instructions, ici en indiquant dans un tableau les valeurs successives prises par les deux variables *a* et *b*.

	a	***b***
Initialisation	1	5
3e ligne	2	5
4e ligne	2	4
5e ligne	4	−5
6e ligne	4	4

▶ Voir à la fin du cahier pour le tableau de correspondance entre les différentes syntaxes.

Remarque L'affectation d'une variable efface toute valeur antérieurement affectée. Si *x* et *y* sont deux variables, la suite consécutive des deux instructions *x* = *y* et *y* = *x* n'a pas pour effet de permuter les contenus des deux variables.

APPLICATION 2

Écrire une suite d'instructions ayant pour effet de permuter les deux variables *x* et *y*.

On utilise une variable auxiliaire *temp* :

temp = *x* on stocke *x* dans *temp*

x = *y* *x* prend la valeur *y*

y = *temp* *y* prend la valeur *temp* égale à la valeur initiale de *x*

Remarque En Python, on peut réaliser des affectations simultanées, par exemple : « *a*,*b*=2,8 », et on peut réaliser l'échange de deux variables sans l'aide de variable auxiliaire : « *a*,*b*=*b*,*a* ».

APPLICATION 3

Écrire un algorithme qui prend en entrée trois nombres et qui renvoie une permutation circulaire de ces trois nombres, c'est-à-dire qui prend en entrée le triplet (*a*, *b*, *c*) et renvoie le triplet (*c*, *a*, *b*).

```
1 a=input("a= ")
2 b=input("b= ")
3 c=input("c= ")
4 temp1=a
5 temp2=b
6 a=c
7 b=temp1
8 c=temp2
9 afficher([a,b,c])
```

Remarques

- L'instruction **input** met l'ordinateur en position d'attente.

La valeur donnée par l'utilisateur sera ensuite affectée à la variable *a*.

- L'instruction **afficher** affiche la valeur de la variable précisée.

4 L'instruction conditionnelle

On est très souvent amenés à effectuer des instructions sous certaines conditions.

Il s'agit, par exemple, d'effectuer des instructions qui dépendent, la plupart du temps, de la comparaison de deux valeurs affectées à deux variables. Ces relations de comparaison sont $<$, $>$, $\leqslant$, $\geqslant$, $=$, $\neq$.

EXEMPLE 1

En regardant le plan de Paris (▶ page 3), le chemin de Yasmine, à partir de la rue de l'Observatoire, pourrait être :

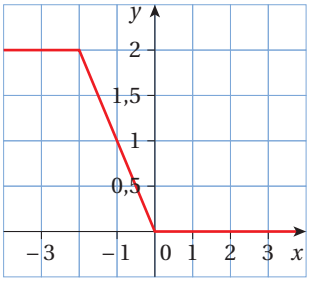
« Continuez jusqu'au bout, tournez à droite, prenez la première à gauche puis :
si la troisième rue à droite est en travaux, **alors** prenez la suivante à droite, puis la première à droite, puis la première à gauche, et continuez tout droit jusqu'à la bifurcation de la rue, vous y êtes.
Sinon, empruntez cette rue, continuez tout droit jusqu'à la bifurcation de la rue, vous y êtes »

Remarque En langage naturel une instruction conditionnelle peut se formuler par : « Si … alors … ; sinon … », ce qui se traduit en langage Python par :

```
if condition :
    instructions
else :
    instructions
```

APPLICATION 4

Voici ci-contre le graphe d'une fonction f définie sur l'intervalle $[-4 ; 4]$.
Écrire un algorithme qui renvoie l'image d'un réel x donné en entrée.




```
def f(x):
    if x>4 or x<-4:
        print("f n'est pas définie")
    elif x>=-4 and x<-2:
        return 2
    elif x<0:
        return -x
    elif x<=4:
        return 0
```

EXEMPLE 3 *La suite de Syracuse*

Considérons l'algorithme en langage naturel :

```
Entrée :                    Un entier naturel a
Traitement et Sortie :      Si a est pair alors on affiche la valeur a÷2
                            Si a est impair alors on affiche la valeur 3a+1
```

Cet algorithme peut, par exemple, s'écrire à l'aide du logiciel AlgoBox ou encore en langages Python ou TI :

AlgoBox

```
1   VARIABLES
2     a EST_DU_TYPE NOMBRE
3   DEBUT_ALGORITHME
4     LIRE a
5     SI (a%2==0) ALORS
6       DEBUT_SI
7       a PREND_LA_VALEUR a/2
8       FIN_SI
9       SINON
10        DEBUT_SINON
11        a PREND_LA_VALEUR 3*a+1
12        FIN_SINON
13    AFFICHER a
14  FIN_ALGORITHME
```

Python*

```
a=int(input("a= "))
if a%2==0:
    a=a/2
else:
    a=3*a+1
print(a)
```

TI

```
PROGRAM:SYRACUSE
:Input "A=",A
:If A-2*ent(A/2)
=0
:Then
:A/2→A
:Else
:3*A+1→A
```

* En Python, « a%2 » désigne le reste de la division euclidienne de l'entier a par 2.

Attention ! Il ne faut pas confondre l'égalité et l'affectation.
En Python, « = » est le symbole d'affectation, « = = » est celui de l'égalité. (▶ voir tableau à la fin du cahier).

Remarque Dans la plupart des langages, les blocs d'instructions doivent être délimités par des symboles spécifiques.
En Python, ces délimiteurs de blocs sont repérés par les sauts de ligne et les indentations (décalage par rapport à la marge). En Python, la ligne d'en-tête d'un bloc d'instructions se termine toujours par « : ».

5 La boucle conditionnelle

On peut être amené à répéter un bloc d'instructions tant qu'une condition reste vérifiée.

En langage naturel une répétition en boucle peut se formuler par : « tant que … ».

Dans la plupart des langages, la syntaxe d'une répétition en boucle est **while** (*tant que* en anglais). Si la condition qui suit la ligne d'en-tête contenant l'instruction **while** est vérifiée, alors le programme exécute toutes les instructions du bloc qui suit, sinon le bloc est entièrement ignoré.

EXEMPLE 1

Le programme Python ci-dessous affiche les carrés et cubes des entiers de 1 à 20.

```
n=1
while n<=20:
    print(n,n**2,n**3)
    n=n+1
```

EXEMPLE 2 ***La « suite de Fibonacci »***

La suite de Fibonacci est une suite de nombres, dont les deux premiers sont égaux à 1 et chaque terme suivant est égal à la somme des deux termes qui le précèdent. Les premiers termes de cette suite sont :

1, 1, 2 (=1+1), 3 (=1+2), 5 (=2+3), 8 (=3+5), 13 (=5+8),...

Le programme ci-dessous affiche les premiers termes de cette suite.

On pourra l'exécuter pas à pas afin de comprendre le rôle de chaque variable.

```
a,b,c=0,1,0
print(b)
while c<20:
    a,b,c=b,a+b,c+1
    print(b)
```

Dans cet exemple, les variables *a* et *b* contiennent à chaque itération deux termes consécutifs de la suite. La variable *c* est un compteur de boucles.

APPLICATION 5

Écrire un algorithme qui affiche la table de multiplication par *n*, *n* entier naturel donné en entrée.

```
n=input("n=")
i=1
while i<=10:
    print(i,'x',n,'=',i*n)
    i=i+1
```

APPLICATION 6

Voici un algorithme écrit en Scilab. Quelle est la valeur de la variable *x* à la fin des instructions ?

```
1 n=10
2 x=0
3 i=0
4 while i<n
5     x=x+i
6     i=i+2
7 end
8
```

La dernière valeur de la variable *x* est 20.

6 La boucle itérative

L'autre moyen de répéter des instructions peut se réaliser à l'aide de l'instruction **for**.

En Python, la boucle **for** parcourt dans l'ordre les éléments d'une séquence quelconque. La syntaxe est la suivante :

```
for variable in séquence :
    opérations à répéter
```

EXEMPLE

Calcul de la somme des entiers de 1 à 200 en langage Python :

```
S=0
for i in range(201):
    S=S+i
print(S)
```

Remarque L'instruction **range**(201) désigne la séquence des entiers 0, 1, 2, ..., 200.

Si a et b sont des entiers (avec $a < b$), **range**(a, b) désigne la séquence des entiers n vérifiant $a \leqslant n < b$.

APPLICATION 7

Écrire un algorithme qui affiche les lignes suivantes :

```
*
**
***
****
```

Puis

```
****
***
**
*
```

```
for i in range(1,5):
    print(i*'*')
```

```
for i in range(1,5):
    print((5-i)*'*')
```

Remarque Sous Python, l'instruction (3*'*') affiche « *** ».

APPLICATION 8

Que réalise cet algorithme écrit en Python ?

```
for i in range(2,10):
    for j in range(1,11):
        print(i,'x',j,'=',i*j)
    print()
```

Ce programme affiche les tables de multiplication des entiers de 2 à 9.

Remarque On peut toujours remplacer une boucle itérative par une boucle conditionnelle :

- Si on connaît le nombre de passages dans la boucle, il suffit de définir une variable compteur qui compte le nombre de passages dans la boucle. Il faudra penser à initialiser correctement ce compteur et à l'incrémenter à l'intérieur de la boucle.
- Si on ne connaît pas le nombre de passages dans la boucle, il faut déterminer la condition d'arrêt et s'assurer que cette dernière sera atteinte.

APPLICATION 9

Écrire un programme qui affiche les tables de multiplication de 2 à 9 en n'utilisant que des boucles conditionnelles.

```
i=2
while i<10:
    j=1
    while j<11:
        print(i,'x',j,'=',i*j)
        j=j+1
    print()
    i=i+1
```

7 Les fonctions

Afin de faciliter la lecture d'un programme complexe, on peut le décomposer en sous-programmes plus simples à lire et interpréter. Ces mêmes sous-programmes peuvent de plus être utilisés plusieurs fois dans le programme initial. Il peut alors être intéressant de définir de nouvelles instructions pour la construction d'un programme.

La syntaxe en Python pour définir une fonction est la suivante :

```
def nomdelafonction (paramètres) :
        ........................
        bloc d'instructions
        ........................
```

Dans la suite du programme, l'appel à cette fonction s'effectuera simplement par son nom suivi des paramètres entre parenthèses, de la même manière qu'un appel à une fonction prédéfinie du langage.

EXEMPLE

La fonction **sec** suivante convertit en secondes une durée exprimée heure, minute et seconde (l'instruction **return** renvoie la valeur calculée par la fonction) :

```
def sec (h, m, s) :
    "conversion en secondes"
    return 3600*h+60*m+s
```

APPLICATION 10

Écrire une fonction qui calcule la somme des n premiers entiers naturels.

En Python :

```
def somme(n):
    S=0
    i=1
    while i<=n:
        S=S+i
        i=i+1
    return S
```

Avec Scilab :

```
function S=somme(n)
    S=0
    i=1
    while i<=n
        S=S+i
        i=i+1
    end
endfunction
```

8 Utilisation de GéoTortue (langage logo)

▶ **voir le site *geotortue.free.fr***

Ce logiciel permet de réaliser des figures géométriques en déplaçant, à l'aide de commandes, le curseur qui laisse une trace derrière lui.

Les principales fonctions de ce module sont les suivantes :

Instruction et syntaxe	Action
av L	Avance d'une longueur L
re L	Recule d'une longueur L
tg a	Tourne à gauche de a degrés
td a	Tourne à droite de a degrés
lc	Lève le crayon traceur
bc	Abaisse le crayon
vg	Vide graphique, efface tous les parcours
tlp x y	Téléporte la tortue au point de coordonnées (x, y)
rep n [bloc]	Répète n fois un bloc de commandes
si (*conditon*) **alors** [*actions*] **sinon** [actions]	Instruction conditionnelle
tant_que (*condition*) **alors** [*actions*]	Boucle conditionnelle

EXEMPLE

La suite des instructions suivantes dessine un carré de côté 100 :

```
> av 100 ; td 90 ; av 100 ; td 90 ; av 100 ; td 90 ; av 100 ; td 90
```

On l'écrira plus simplement :

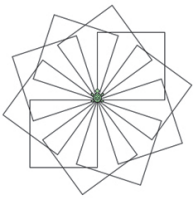
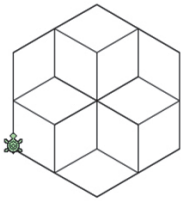
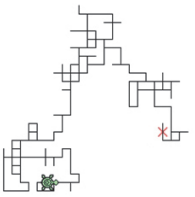
```
> rep 4  [ av 100 ; td 90 ]
```

On peut créer de nouvelles procédures qui pourront être appelées ultérieurement dans d'autres programmes. Définissons par exemple la procédure **carre** qui prendra la longueur du côté en argument :

```
1 > pour carre c
2 > rep 4  [ av c ; td 90 ]
3 > fin
```

L'exécution de **carre** 100 dessinera un carré de côté 100.

On peut ainsi réaliser des dessins géométriques plus complexes :

<table>
<tr>
<td>  </td>
<td>  </td>
<td>  </td>
</tr>
<tr>
<td>

```
rep 10  [ carre 100 ; td 36 ]
```

</td>
<td>

```
pour hexagone c
rep 6  [ av c ; td 60 ]
fin

rep 6  [ av 50; hexagone 50 ; av 50 ; td 60]
```

</td>
<td>Une marche aléatoire :

```
pour dep n
rep n  [x:=alea(4); si (x==1) alors [av 10]; si (x==2) alors [tg 90 ; av 10  ] ;
si (x==3) alors [ td 90 ; av 10] ;si (x==4) alors [ re 10 ]]
fin
```

</td>
</tr>
</table>

APPLICATION 11

Réaliser les figures suivantes :

Fig. 1

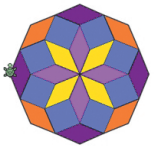


Fig. 2

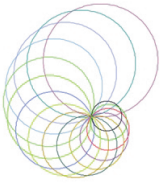


Fig. 3

COUP DE POUCE

- Dans la figure 1, on pourra utiliser une procédure **dodecagone** et une procédure **carre**.
- Dans la figure 2, on pourra par exemple observer que la figure est constituée du motif ci-contre, composé de deux losanges que l'on a fait tourner huit fois. Il faudra bien entendu être attentif à la position de la tortue. Pour la gestion des couleurs, on se référera au site du programme GéoTortue.
- La figure 3 est constituée de cercles. Un cercle pourra être approché par un polygone régulier ayant un nombre important de côtés.

Fig. 1 :

```
pour dodecagone c
rep 12  [ av c ; td 30 ]
fin

pour carre c
rep 4  [ av c ; td 90 ]
fin

pour etoile
· dodecagone 50
· rep 6  [ td 30 ; carre 50 ; av 50 ; tg 30 ; av 50 ; td 60]
· fin
```

Fig. 2 :

```
pour losange
rep 2  [ av 40 ; td 45 ; av 40 ; td 135]
fin

pour  motif
losange
td 45 ; av 40 ; td 45
losange
tg 45 ; re 40 ; tg 45
fin

pour fig2
td 22.5
rep 8  [av 40 ; motif ; av 40 ; td 45 ]
fin
```

Fig. 3 :

```
pour cercle t
rep 180  [ av t ; td 2 ]
fin
pour spi
t:=1
rep 15  [ cercle t ; td 20 ; t:=t+0.2; couleur alea(255) alea(255) alea(255)]
fin
```

Calcul algébrique

1

Prérequis : Expressions algébriques

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Identifier les données d'entrée, de sortie, le traitement

NOTION
- Affectation, variable

D'après brevet des collèges 2008

On donne le programme de calcul suivant :

```
Choisir un nombre
Multiplier ce nombre par 3
Ajouter le carré du nombre choisi
Multiplier par 2
Écrire le résultat
```

1 Calculer la valeur exacte du résultat dans les cas ci-dessous.

a. Le nombre choisi est –5. 20

b. Le nombre choisi est $\sqrt{5}$. $6\sqrt{5}+10$

2 Écrire ci-dessous, dans la syntaxe de votre choix, le programme précédent qui prend en entrée un nombre A et qui renvoie le nombre calculé.

```
Demander un nombre A
Résultat prend la valeur 3A
Résultat prend la valeur 3A + A²
Résultat prend la valeur 6A + 2A²
Afficher résultat
```

3 Quels nombres peut-on choisir pour que le résultat obtenu soit 0 ?

Il suffit de résoudre l'équation $6x + 2x^2 = 0$, dont les solutions sont 0 et –3.

2

Prérequis : Opérations sur les fractions, fractions irréductibles

COMPÉTENCES ALGORITHMIQUES
- Modifier un algorithme
- Identifier les données d'entrée, de sortie, le traitement

NOTION
- Boucle conditionnelle

On souhaite écrire un algorithme permettant de réduire la somme $\frac{a}{b}+\frac{c}{d}$ au même dénominateur, a, b, c et d étant des entiers.

1 Pour obtenir un résultat exact, on introduit deux variables u et v telles que $\frac{a}{b}+\frac{c}{d}=\frac{u}{v}$; u et v seront donc les sorties de cet algorithme.

a. Donner les formules permettant de calculer u et v.

$u = ad + bc$ et $v = bd$.

b. Écrire l'algorithme.

```
Entrer 4 entiers a, b, c et d
u prend la valeur ad + bc
v prend la valeur bd
Afficher u
Afficher v
```

2 Quelles sont les contraintes sur les entiers b et d ?

Les nombres b et d doivent être non nuls.

3 Modifier le programme précédent afin qu'il renvoie un message d'erreur lorsque ces contraintes ne sont pas respectées.

```
Entrer 4 entiers a, b, c et d
Si b = 0 alors
    Afficher « une des fractions n'est pas définie »
Fin si
Si d = 0 alors
    Afficher « une des fractions n'est pas définie »
Fin si
u prend la valeur ad + bc
v prend la valeur bd
Afficher les valeurs de u et v
```

4 On souhaiterait enfin que le programme renvoie le résultat de la somme sous la forme d'une fraction irréductible. On suppose disposer d'une fonction qui calcule le PGCD de deux entiers. Compléter alors le programme précédent afin qu'il renvoie la fraction sous forme irréductible.

Après les calculs de u et v, on rajoute les instructions :

```
delta prend la valeur PGCD(u, v)
u prend la valeur u/delta
v prend la valeur v/delta
```

3

Prérequis : Racine carrée, aire du rectangle, valeur absolue d'un nombre réel

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Valider la solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

- Boucle itérative
- Boucle conditionnelle

L'algorithme de Héron permet de déterminer des valeurs approchées de racines carrées d'entiers.
Pour les mathématiciens grecs, déterminer une valeur approchée de $\sqrt{2}$ revient à construire un carré dont l'aire est 2. Pour ce faire, partons d'un rectangle de longueur 2 et de largeur 1. Son aire est 2, mais ce n'est pas un carré.
On construit alors un deuxième rectangle d'aire égale à 2 et dont la longueur est la moyenne arithmétique des côtés du rectangle précédent, soit $\frac{1+2}{2} = \frac{3}{2}$. Sa largeur est alors égale à $\frac{4}{3}$.
On réitère le processus en construisant une suite de rectangles d'aire 2 et dont les longueurs sont égales aux moyennes des dimensions du rectangle précédent.

1 Compléter le tableau suivant avec des valeurs approchées des dimensions des rectangles successifs :

Rang de l'itération	Longueur du 1er côté	Longueur du 2e côté
1	2	1
2	$\frac{3}{2}$	$\frac{4}{3}$
3	1,416	1,411
4	1,414 215	1,414 211
5	1,414 211 356 237 31	1, ,414 211 356 237 31

Que peut-on conjecturer ?

Les rectangles convergent vers le carré de côté $\sqrt{2}$.

2 Nous voulons écrire un algorithme qui calcule les valeurs successives des longueurs des rectangles.

a. L'algorithme précédent peut s'écrire de la façon suivante :

```
Entrer un entier N
a prend la valeur 2
b prend la valeur 1
Pour i allant de 1 jusqu'à N
    a prend la valeur (a + b)/2
    b prend la valeur 2/a
    Afficher a et b
Fin pour
```

Traduire cet algorithme dans une syntaxe au choix et vérifier les valeurs obtenues à la question **1**.

En Scilab :

```
1 N=input("entrer le nombre d''itérations ")
2 a=2
3 b=1
4 for i=1:N
5     a=(a+b)/2
6     b=2/a
7     afficher(a,b)
8 end
```

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

b. On admet qu'au fur et à mesure des itérations, les rectangles « tendent » vers un carré d'aire 2, donc de côté $\sqrt{2}$. Modifier le programme précédent pour que les itérations s'arrêtent dès que la valeur absolue de la différence entre ses dimensions est inférieure à une précision donnée en entrée.

```
1 a=2
2 b=1
3 precision=0.0000000001
4 while abs(a-b)>precision
5     a=(a+b)/2
6     b=2/a
7     afficher(a,b)
8 end
```

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

3 À l'aide de ce nouveau programme, déterminer une valeur approchée de $\sqrt{11}$ à 10^{-13} près.

En partant cette fois d'un rectangle 1 sur 11, on obtient la valeur approchée 3,316 624 790 355 4 à 10^{-13} près à la 6e itération de l'algorithme.

4

Prérequis : Division euclidienne, partie entière (définie dans l'énoncé)

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Mettre au point une solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

- Boucle conditionnelle
- Instruction conditionnelle

Voici un algorithme qui définit une suite de nombres :

```
Prendre un entier naturel A
Tant que A ≠ 4, effectuer les instructions suivantes :
    Si A se termine par le chiffre 4, barrer ce chiffre
    Si A se termine par le chiffre 0, barrer ce chiffre
    Sinon multiplier A par 2
Fin tant que
Afficher A
```

1 Effectuer cet algorithme pas à pas en prenant différentes valeurs initiales de A.

Pour $N = 15$, par exemple, l'algorithme renvoie la suite : 30 ; 3 ; 6 ; 12 ; 24 ; 2 ; 4.

2 **a.** Si A est un entier naturel, par quel calcul peut-on obtenir le chiffre des unités ?

COUP DE POUCE

On pourra utiliser la fonction « partie entière ». La partie entière d'un nombre réel x est l'entier, noté $E(x)$ tel que : $E(x) \leqslant x < E(x) + 1$. La partie entière d'un réel x est donc le plus grand entier inférieur ou égal à x.

Le chiffre des unités est le reste dans la division euclidienne par 10, c'est donc la partie entière de $\frac{x}{10}$: $r = x - 10 \times E\left(\frac{x}{10}\right)$.

b. Si A est un entier naturel, par quel calcul obtient-on l'entier privé de son chiffre des unités ?

$\frac{x-r}{10}$

c. Écrire un programme qui prend un entier A en entrée et renvoie la suite des entiers définie par l'algorithme de l'énoncé.

En Scilab :

```
A=input("A= ")
while A<>4
    r=reste(A,10)
    if r==4 then
        A=(A-r)/10
        afficher(A)
    elseif r==0 then A=(A-r)/10
        afficher(A)
    else A=A*2
        afficher(A)
    end
end
```

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

5

Prérequis : Calcul algébrique

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Mettre au point une solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTION

- Boucle conditionnelle

Voici un programme écrit à l'aide du logiciel Scilab :

```
function y=f(x)
    a=1
    b=1
    y=0
    while a<=x
        y=y+1
        b=b+2
        a=a+b
    end
endfunction
```

1 Exécuter ce programme avec les 20 premiers entiers naturels. Que peut-on conjecturer sur la valeur renvoyée par le programme ?

$f(1) = f(2) = f(3) = 1$; $f(4) = f(5) = f(6) = f(7) = f(8) = 2$;

$f(9) = f(10) = \ldots = f(15) = 3$;

$f(16) = \ldots = f(20) = 4$.

La fonction semble renvoyer le plus grand entier dont le carré est inférieur à l'argument donné.

2 En étudiant l'évolution des différentes variables, expliquer la conjecture observée.

Itération		1	2	3	...	n
y	0	1	2	3	...	n
b	1	3	5	7	...	$2n + 1$
a	1	4	9	16	...	n^2

Généralités sur les fonctions

6

Prérequis : Calcul approché

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Valider un programme simple

NOTION
- Affectation, variable

Un algorithme est saisi sur une calculatrice.

```
PROGRAM:IDEM
:Input "X ? ",X
:Input "A ? ",A
:X+A→X
:X-A→X
:Disp "X = ",X
```

Les écrans n° 1 et n° 2 donnent l'affichage après son exécution pour deux valeurs différentes en entrée.

Ecran n°1 :

```
prgmIDEM
X ? 7
A ? 3
X =
                7
            Done
```

Ecran n°2 :

```
prgmIDEM
X ? 10^-80
A ? 45
X =
                0
            Done
```

a. Pour cet algorithme, indiquer les valeurs successives de la variable X. Quel écran est conforme à la sortie attendue ?

Les valeurs successives de la variable X sont X, $X + A$ et $(X + A) - A = X$.

La valeur attendue en sortie est donc la valeur entrée pour X. Pour $X = 7$, l'écran n° 1 est conforme aux attentes.

b. En regardant le mode d'emploi de la calculatrice utilisée, on constate qu'elle ne permet que des calculs avec des nombres décimaux ayant moins de 15 décimales.
Expliquer alors l'affichage qui semble non conforme.

Dans le cas affiché par l'écran n° 2, la calculatrice arrondit la somme $X + A$ à 45 car elle n'arrive pas à stocker en mémoire la valeur 45,000.....0001.

La différence $(X + A) - A$ donne donc $45 - 45 = 0$, ce qui explique pourquoi la valeur affichée en sortie n'est pas égale à 10^{-80}.

7

Prérequis : Tableau de valeurs d'une fonction

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Adapter un algorithme aux contraintes du langage de programmation

NOTION
- Boucle itérative

On donne deux algorithmes programmés en Xcas :

Algorithme n°1 :

```
valeurs():={
  x:=0
  pour i de 1 jusque 4 faire
    x:=2+2*i
    y:=x^2-x/5
    afficher(x,y)
  fpour;
}:;
```

Algorithme n°2 :

```
valeurs():={
  x:=0
  pour i de 1 jusque 4 faire
    x:=x+i
    y:=x^2-x/5
    afficher(x,y)
  fpour;
}:;
```

a. Dans les tableaux ci-dessous, indiquer les valeurs affichées pour les variables x et y dans les 4 boucles.

Algorithme n° 1				
i	1	2	3	4
x	4	6	8	10
y	15,2	34,8	62,4	98

Algorithme n° 2				
i	1	2	3	4
x	1	3	6	10
y	0,8	8,4	34,8	98

b. Programmer le deuxième de ces algorithmes à la calculatrice pour vérifier les résultats annoncés. Puis recopier ci-dessous le programme saisi.

```
PROGRAM:VALEURS
:For(I,1,4)
:X+I→X
:X²-X/5→Y
:Disp X,Y
:Pause
:End
```

8

Prérequis : Pourcentages et coefficient multiplicateur

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Adapter un algorithme aux contraintes du langage de programmation

NOTION
- Boucle itérative

On souhaite calculer le prix d'un kilo de mangues après 10 augmentations successives.

1 **a.** **Cas n°1** : ces augmentations successives sont de 5 %.

Compléter l'algorithme suivant pour qu'il permettre de répondre à la question posée.

```
Entrer un réel positif p
Pour i allant de 1 jusqu'à 10
    | p prend la valeur 1,05 × p
Fin pour
Afficher p
```

b. **Cas n°2** : ces augmentations successives s'élèvent à un montant fixe de 0,30 €.

Écrire l'algorithme qui permet de répondre à la question posée.

```
Entrer un réel positif p
Pour i allant de 1 jusqu'à 10
    | p prend la valeur p + 0,3
Fin pour
Afficher p
```

2 Programmer les deux algorithmes sur la calculatrice et indiquer, dans chaque cas, le prix d'un kilo de mangues, d'un prix initial de 5 €, après les 10 augmentations.

Prix du kilo de mangues

après 10 augmentations :

- dans le cas n°1 : environ 8,15 € ;
- dans le cas n°2 : 8 €.

```
PROGRAM:PRIX
:Input "P ? ",P
:P→Q
:For(I,1,10)
:1.05*P→P
:Q+0.3→Q
:End
:Disp P,Q
```

```
prgmPRIX
P ? 5
      8.144473134
                 8
             Done
```

9

Prérequis : Variations et maximum d'une fonction

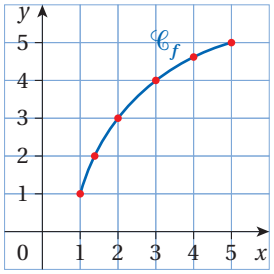
COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Identifier les données d'entrée, de sortie, le traitement
- Valider la solution algorithmique

NOTIONS

- Instruction conditionnelle
- Boucle conditionnelle

a. Soit f la fonction définie sur l'intervalle [1 ; 5] dont la courbe représentative est donnée ci-dessous.



On considère l'algorithme suivant :

```
x prend la valeur 1
Tant que x < 5 et f(x + 1) > f(x)
    x prend la valeur x + 1
Fin tant que
Afficher x
```

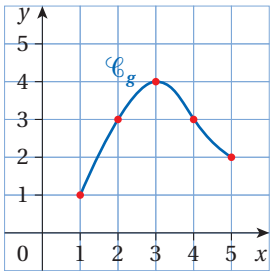
Expliquer pourquoi cet algorithme affiche 5 en sortie.

D'après la représentation graphique, l'inégalité $f(x + 1) > f(x)$ est vraie pour x valant 1, 2, 3 ou 4.

À l'entrée de la boucle, les valeurs successives de la variable x seront donc 1, 2, 3, 4.

À la dernière sortie de la boucle, la variable x vaudra donc 5.

b. Soit g la fonction définie sur l'intervalle [1 ; 5] dont la courbe représentative est donnée ci-dessous.



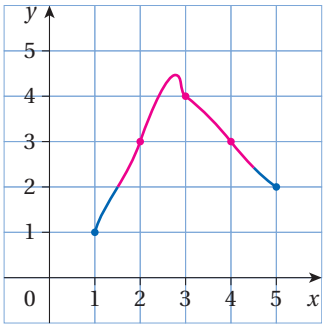
On considère l'algorithme suivant :

```
x prend la valeur 1
y prend la valeur 5
Tant que x < 5 et g(x) < g(x + 1)
    x prend la valeur x + 1
Fin tant que
Tant que y > 1 et g(y – 1) > g(y)
    y prend la valeur y - 1
Fin tant que
Si x = y alors
    Afficher « Le maximum de la fonction g sur [1 ; 5] est atteint en »
    Afficher x
Sinon
    Afficher « Le maximum de la fonction g sur [1 ; 5] n'est pas atteint en »
    Afficher x
Fin si
```

Qu'affichera ce deuxième algorithme pour la fonction g ?

L'algorithme affichera : « Le maximum de la fonction g est atteint en 3 ».

c. On considère la courbe incomplète d'une fonction définie sur l'intervalle [1 ; 5] donnée ci-dessous. Compléter cette représentation graphique pour que l'algorithme précédent, appliqué à cette fonction, affiche une affirmation fausse.



Avant l'affichage, x et y valent 3, donc l'algorithme affichera que le maximum de la fonction est atteint en 3 ; ceci est faux car le maximum de cette fonction semble atteint entre 2,5 et 3 strictement.

10

Prérequis : Image d'un nombre par une fonction

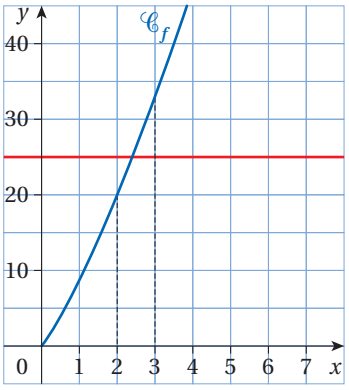
COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Valider un programme simple

NOTIONS

- Affectation, variable
- Boucle conditionnelle

Soit la fonction f définie sur $[0 ; +\infty[$ par $f(x) = x(x + 8)$.
On veut trouver le plus petit entier naturel x tel que $f(x)$ soit plus grand qu'un nombre A que l'on choisit.
Par exemple si $A = 25$, le nombre x qui convient est $x = 3$ car $f(0) = 0$, $f(1) = 9$ et $f(2) = 20$ sont inférieurs à 25, alors que $f(3) = 33$ est supérieur à 25.



On propose l'algorithme suivant :

```
Entrer un réel positif A
x prend la valeur 0
y prend la valeur 0
Tant que y ⩽ A
    x prend la valeur x + 1
    y prend la valeur x(x + 8)
Fin tant que
Afficher x, y
```

a. Compléter le tableau ci-dessous des valeurs successives de x et y en faisant fonctionner l'algorithme pour $A = 100$.

x	0	1	2	3	4	5	6	7	
y	0	9	20	33	48	65	84	105	

b. À l'aide d'un graphique de f tracé sur la calculatrice, déterminer la valeur de x en sortie pour $A = 500$ en entrée.

La valeur de x sera 97 car $f(96) = 9\,984$ et $f(97) = 10\,185$.

11

Prérequis : Représentation graphique d'une fonction

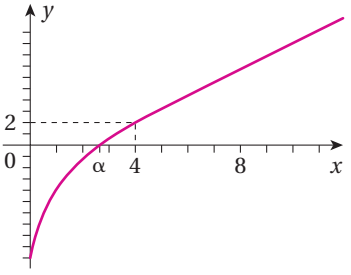
COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Identifier les données d'entrée, de sortie, le traitement

NOTIONS

- Instruction conditionnelle
- Boucle conditionnelle

La fonction f donnée par $f(x) = x - \dfrac{10}{x+1}$ est croissante sur l'intervalle $[0 ; 8]$ et l'équation $f(x) = 0$ admet une unique solution α comme l'illustre sa représentation graphique ci-contre.



On souhaite encadrer la valeur α avec la précision p choisie.
Le principe est le suivant :

- on commence par calculer l'image du centre de l'intervalle $[0 ; 8]$;
- 4 est le centre de l'intervalle $[0 ; 8]$ et $f(4) = 2$ est positif, donc $\alpha < 4$ car la fonction f est croissante ;
- on peut en conclure que $\alpha \in [0 ; 4]$;
- on réitère le processus sur l'intervalle $[0 ; 4]$.

a. Par quelle formule calcule-t-on le centre d'un intervalle $[a ; b]$ à partir des bornes a et b ?

Le centre de l'intervalle $[a ; b]$ est $\dfrac{b-a}{2}$.

b. Compléter le tableau suivant pour obtenir un encadrement d'amplitude 0,5. On utilisera une calculatrice pour obtenir les valeurs des images $f(x)$.

Borne a	Borne b	$b-a$ (amplitude de l'encadrement)	x (centre de l'intervalle)	$f(x)$ (image du centre)	Signe de l'image du centre
0	8	8	4	2	positif
0	4	4	2	$\approx -1{,}333\,3$	négatif
2	4	2	3	0,5	positif
2	3	1	2,5	$\approx -0{,}357\,1$	négatif
2,5	3	0,5	2,75	$\approx 0{,}083\,33$	positif

c. L'algorithme ci-contre donne l'encadrement souhaité.

```
Entrer un nombre positif p
a prend la valeur 0
b prend la valeur 8
Tant que b − a > p
    Si f((b − a)/2) ⩾ 0 alors
        b prend la valeur (b − a)/2
    Sinon
        a prend la valeur (b − a)/2
    Fin si
Fin tant que
Afficher a, b
```

Quel sera l'affichage donné en sortie par cet algorithme lorsque l'entrée p est 0,1 ?

Borne a	Borne b	$b-a$ (amplitude de l'encadrement)	x (centre de l'intervalle)	$f(x)$ (image du centre)	Signe de l'image du centre
2,5	3	0,5	2,75	$\approx 0{,}083\,33$	positif
2,5	2,75	0,25	2,625	$\approx -1{,}333\,6$	négatif
2,625	2,75	0,125	2,687 5	$\approx -0{,}024\,3$	négatif
2,687 5	2,75	0,062 5			

Le calcul s'arrête lorsque l'amplitude vaut 0,062 5 et on aura $a = 2{,}687\,5$ et $b = 2{,}75$.

Fonctions de référence

12

Prérequis : Calcul de pourcentages

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Adapter un algorithme aux contraintes du langage de programmation

NOTION
- Instruction conditionnelle

L'affiche d'un marchand de bonbons indique :

- 2 € les 100 g jusqu'à un kilogramme ;
- 1,50 € les 100 g supplémentaires au-delà de 1000 g ;
- 15 % de réduction pour toute commande d'un montant supérieur à 30 €.

L'algorithme suivant était programmé dans la caisse enregistreuse pour indiquer le prix à payer par un client.

```
Entrer un nombre positif m
n prend la valeur m/100
Si n ⩽ 10 alors
    p prend la valeur 2 × n
Fin si
Si n > 10 alors
    p prend la valeur 2 × 10 + 1,5 × (n - 10)
Fin si
Si p > 30 alors
    p prend la valeur 0,85 × p
Fin si
Afficher p
```

a. Remettre les nombres qui ont été effacés sur l'affiche.

b. Programmer l'algorithme pour compléter le tableau suivant :

Masse achetée (en g)	1500	1600	1700	1800	1900
Pris (en €)	27,50	29,00	25,92	27,20	28,46

```
1  m= input('Masse en g de bonbons achetés : ');
2  n=m/100
3  if n<=10 then
4      p=2*n;
5  end
6  if n>10 then
7      p=2*10+1.5*(n-10);
8  end
9  if p>30 then
10     p=0.85*p;
11 end
12 disp(p, "Le prix à payer est :");
```

c. Un client a payé 26 €. Quelle masse de bonbons a-t-il pu acheter ?

Il a pu acheter 1400 g mais également 1706 g.

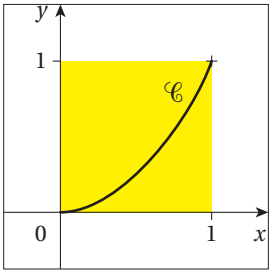
13

Prérequis : Fonction carré

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Mettre au point une solution algorithmique

NOTIONS
- Affectation, variable
- Instruction conditionnelle
- Boucle itérative

Soit f la fonction définie sur $[0 ; 1]$ par $f(x) = x^2$ et $\mathscr{C}$ sa courbe représentative dans un repère orthonormal. On souhaite simuler un jeu de fléchettes, où l'on vise au hasard un point quelconque du carré jaune.
Pour ceci, on considère le point d'impact M de la fléchette qui a comme coordonnées x et y. Chacune de ces coordonnées est un nombre réel, choisi de manière aléatoire dans l'intervalle $[0 ; 1]$.



a. Écrire un algorithme qui simule le jeu précédent et renvoie la position du point d'impact M par rapport à la courbe $\mathscr{C}$.

```
x prend la valeur d'un nombre choisi de manière aléatoire dans [0 ; 1]
y prend la valeur d'un nombre choisi de manière aléatoire dans [0 ; 1]
Si x² < y alors
    Afficher « Le point M(x ; y) est au-dessous de C.»
Fin si
```

b. Dans le programme ci-contre, écrit en Scilab, l'algorithme de la question **a** a été modifié.
Que représente la valeur affichée par cet algorithme ?

La valeur affichée est la proportion du nombre de points situés en dessous de la courbe lorsque l'on a effectué 100 tirs de fléchette.

```
n=0
for i=1:100
    x=rand()
    y=rand()
    if x^2<y then
        n=n+1
    end
end
disp(n/100)
```

14

Prérequis : Fonction inverse, équation d'une droite

COMPÉTENCES ALGORITHMIQUES

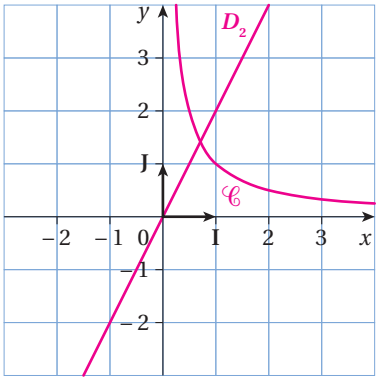
- Mettre au point une solution algorithmique
- Valider la solution algorithmique

NOTION

- Instruction conditionnelle

Soit f la fonction définie sur l'intervalle $]0 ; +\infty[$ par $f(x) = \dfrac{1}{x}$ et $\mathscr{C}$ sa courbe représentative dans un repère (O ; I , J). Soit a un nombre réel donné et D_a la droite d'équation $y = ax$ dans le repère (O ; I , J).

a. Dans le repère ci-dessous, tracer $\mathscr{C}$ et D_2.



b. Écrire un algorithme qui, pour une entrée a, donne le nombre de points d'intersection de $\mathscr{C}$ avec D_a. Expliquer, en prenant quelques exemples, comment fonctionne l'algorithme proposé.

```
Entrer un nombre a
Si a > 0 alors
    Afficher « Il y a un point d'intersection »
Sinon
    Afficher « Il n'y a aucun point d'intersection »
Fin si
```

• Pour $a = -8$, l'affichage « aucun point » est exact car l'équation $\dfrac{1}{x} = -8x$ (équivalente à $x^2 = -\dfrac{1}{8}$) n'a pas de solution dans l'intervalle $]0 ; +\infty[$.

• Pour $a = 8$, l'affichage « un point » est exact car l'équation $\dfrac{1}{x} = 4x$ (équivalente à $x^2 = \dfrac{1}{4}$) a une de ses solutions dans l'intervalle $]0 ; +\infty[$.

15

Prérequis : Fonction définie par intervalles

COMPÉTENCES ALGORITHMIQUES

- Identifier les données d'entrée, de sortie, le traitement
- Mettre au point une solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTION

- Instruction conditionnelle

Un site de développement de photos affiche les tarifs suivants :
• de 1 à 25 tirages : 0,15 € par photo et 3 € de frais de port ;
• de 26 à 70 tirages : 0,10 € par photo et 4 € de frais de port ;
• au-delà de 70 tirages : 0,05 € par photo et 7 € de frais de port.

a. Compléter l'algorithme suivant pour qu'il donne le prix à payer quel que soit le nombre de photos commandées.

```
Entrer un entier n supérieur ou égal à 1
Si n ⩽ 25 alors
    p prend la valeur 3 + 0,15n
Fin si
Si n ⩾ 26 et n ⩽ 70 alors
    p prend la valeur 4 + 0,1n
Fin si
Si n ⩾ 71 alors
    p prend la valeur 7 + 0,05n
Fin si
Afficher p
```

b. Programmer cet algorithme à la calculatrice ou dans un langage de programmation, et recopier ci-dessous cet algorithme.

```
======PHOTOS  ======
"N "?→N
If N≤25
Then 3+0.15×N→P
IfEnd
If N≥26 And N≤70
Then 4+0.1×N→P
IfEnd
If N≥71
Then 7+0.05×N→P
IfEnd
P
```

16

Prérequis : Calcul de pourcentages par coefficients

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Mettre au point une solution algorithmique

NOTION

- Boucle itérative

Le 1er janvier 2013, la dirigeante d'une société promet à ses actionnaires une diminution des coûts de production de 5 % par mois.
Pour les convaincre de la soutenir, elle met en œuvre l'algorithme suivant :

```
Entrer un nombre positif C
Entrer un nombre entier positif N
CF prend la valeur C
Pour I allant de 1 à N
    CF prend la valeur 0,95 × CF arrondie à deux décimales
    Afficher le couple (I ; CF)
Fin pour
```

a. Pour une entrée $C = 100$ et $N = 5$, compléter le tableau ci-dessous qui donne les valeurs affichées par l'algorithme.

I	1	2	3	4	5
CF	95	90,25	85,74	81,45	77,38

b. Proposer un algorithme qui affiche le nombre de mois nécessaires pour diviser par deux les coûts de production.

```
K prend la valeur 1
I prend la valeur 0
Tant que K > 0,5
    I prend la valeur I + 1
    K prend la valeur 0,95 × K
Fin tant que
Afficher le nombre I
```

17

Prérequis : Puissance entière

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Identifier les données d'entrée, de sortie, le traitement

NOTION
- Boucle conditionnelle

L'algorithme ci-contre, écrit en Python, renvoie une valeur y pour tout couple $(x ; n)$ où n est un entier naturel.

```
def f(x,n):
    i=n
    y=1
    while i>0:
        y=y*x
        i=i-1
    return y
```

a. Indiquer dans le tableau ci-dessous la valeur renvoyée pour chacun des couples $(x ; n)$ correspondant.

Entrée x	2	5	3
Entrée n	0	2	3
Sortie y	1	25	27

b. Quelle est l'expression de la valeur renvoyée par l'algorithme en fonction de x et de n ? Justifier la réponse.

La valeur renvoyée est x^n, car à chaque passage dans la boucle le résultat est multiplié par y ; et il y a n passages car i prend successivement les valeurs n, $n - 1$, $n - 2$, … , 2, 1.

18

Prérequis : Calcul algébrique

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Identifier les données d'entrée, de sortie, le traitement
- Mettre au point une solution algorithmique

NOTION
- Boucle conditionnelle

L'algorithme ci-dessous, écrit en Python, renvoie une valeur y pour tout n entier naturel.

```
def f(n):
    i=n
    y=0
    while i>0:
        y=y+n
        i=i-1
    return y
```

a. Expliquer pourquoi cet algorithme donne, sans effectuer un produit, le carré de l'entier n.

Le carré est obtenu car à chacun des n passages dans la boucle on ajoute n à y (n au départ vaut 0).

b. Proposer un algorithme qui calcule le cube d'un entier n, toujours sans effectuer de produit.

```
def f(n):
    i=n
    y=0
    while i>0:
        y=y+n
        i=i-1
    i=n
    z=0
    while i>0:
        z=z+y
        i=i-1
    return z
```

Il suffit cette fois-ci d'ajouter n fois la valeur déjà obtenue en y ($n^2 + n^2 + \ldots + n^2 = n^3$ lorsqu'il y a n termes à additionner).

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

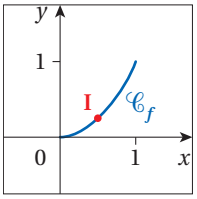
19

Prérequis : Fonction carré, longueur d'un segment dans un repère

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS
- Affectation, variable
- Boucle itérative

Dans le repère orthonormal ci-dessous, la courbe tracée est celle de la fonction f définie sur l'intervalle $[0 ; 1]$ par $f(x) = x^2$.



La distance entre les points A et B, avec $A(x_A ; y_A)$ et $B(x_B ; y_B)$, est égale à $\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$.

a. Entourer, parmi les valeurs proposées, la distance OI avec $I\left(\frac{1}{2} ; \frac{1}{4}\right)$.

• 0,25 • 0,5 • $\frac{\sqrt{5}}{4}$ • $\sqrt{2}$

b. On donne l'algorithme suivant :

Entrer un entier n supérieur ou égal à 1
a **prend** la valeur 0
x_1 **prend** la valeur 0
x_2 **prend** la valeur $x_1 + \frac{1}{n}$
Pour i allant **de** 1 **à** n
  y_1 **prend** la valeur $(x_1)^2$
  y_2 **prend** la valeur $(x_2)^2$
  a **prend** la valeur $a + \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
  x_1 **prend** la valeur x_2
  x_2 **prend** la valeur $x_1 + \frac{1}{n}$
Fin pour
Afficher a

Pour une entrée de n qui vaut 2, compléter le tableau ci-dessous qui indique les valeurs des variables en fin de boucle.

i	y_1	y_2	x_1	x_2	a
1	0	0,25	0,5	1	$\frac{\sqrt{5}}{4} \approx 0{,}559$
2	0,25	1	1	1,5	$\frac{\sqrt{5} + \sqrt{13}}{4} \approx 1{,}46$

c. Programmer cet algorithme, puis donner la valeur affichée pour a lorsque $n = 10$.

```
from math import sqrt
n=int(input("Entrer n "))
a=0
x1=0
x2=x1+1/n
for i in range(1,n+1):
    y1=x1**2
    y2=x2**2
    a=a+sqrt((x2-x1)**2+(y2-y1)**2)
    x1=x2
    x2=x1+1/n
print("a vaut : ",a)
```

On trouve : $a \approx 1{,}478$.

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

Fonctions du second degré et homographiques

20

Prérequis : Maximum d'une liste

COMPÉTENCE ALGORITHMIQUE
- Mettre au point une solution algorithmique

NOTIONS
- Instruction conditionnelle
- Boucle itérative

Une entreprise produit des appareils électroménagers. Le bénéfice réalisé par la fabrication et la vente de x objets est égal à :

$$B(x) = -3x^2 + 76x - 100 \text{ pour } x \text{ appartenant à l'intervalle } [5 ; 20].$$

a. Proposer un algorithme qui permet de trouver le nombre d'appareils à produire pour que le bénéfice de l'entreprise soit maximal.

```
b prend la valeur 0
Pour i allant de 5 à 20
    y prend la valeur B(i)
    Si y > b alors
        x prend la valeur i
        b prend la valeur y
    Fin si
Afficher x et b
```

b. Programmer cet algorithme.

```
Define benef_max()=
Prgm
bm:=0
For i,5,20
 y:=-3·i^2+76·i-100
 If y>bm Then
  x:=i
  bm:=y
 EndIf
EndFor
Disp "Nombre",x,"Benefice max",bm
EndPrgm
```

Le bénéfice est maximal pour $x = 13$ et vaut 381.

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

21

Prérequis : Image par une fonction

COMPÉTENCE ALGORITHMIQUE
- Mettre au point une solution algorithmique

NOTIONS
- Instruction conditionnelle
- Boucle itérative

La fonction f est définie sur $\mathbb{R}\setminus\{-50\}$ par $f(x)=\dfrac{50}{x-50}$.

a. Proposer un algorithme qui donne le nombre d'entiers dans [0 ; 100] dont l'image par f est un entier.

```
n prend la valeur 0
Pour i allant de 0 à 100
    Si i ≠ 50 alors
        y prend la valeur 50/(i − 50)
        Si y entier alors
            n prend la valeur n + 1
        Fin si
    Fin si
Fin pour
Afficher n
```

b. Programmer cet algorithme.

```
from math import *
n=0
for i in range (0,101) :
    if i!=50 :
        y=50./(i-50)
        if floor(y)==y :
            n=n+1
print (n)
```

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

22

Prérequis : Variations d'une fonction polynôme de degré 2

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Mettre au point une solution algorithmique

NOTION
- Instruction conditionnelle

Soit f la fonction définie sur $\mathbb{R}$ par $f(x)=ax^2+bx+c$ où a, b et c sont des nombres réels.

a. Compléter les tableaux de variations suivants :

• $a=1$, $b=-4$ et $c=3$

x		2	
f	↘		↗

• $a=-1$, $b=4$ et $c=3$

x		2	
f	↗		↘

b. L'algorithme suivant donne-t-il une sortie toujours fausse ? Toujours vraie ? Expliquer.

```
Entrer le nombre a
Si a = 0 alors
    Afficher « f est une fonction affine »
Sinon
    Afficher « f est décroissante puis croissante »
Fin si
```

L'algorithme donne une sortie vraie si $a \geqslant 0$ mais une sortie fausse si $a < 0$.

c. Modifier l'algorithme de la question précédente pour qu'il donne des informations sur les variations de f dans les différents cas possibles selon les coefficients a, b ou c.

```
Entrer les nombres réels a et b
Si a = 0 alors
    Si b > 0 alors
        Afficher « f est affine croissante »
    Sinon
        Afficher « f est affine décroissante »
    Fin si
Sinon
    Si a > 0 alors
        Afficher « f est décroissante puis croissante »
    Sinon
        Afficher « f est croissante puis décroissante »
    Fin si
Fin si
```

23

Prérequis : Position relative de courbes

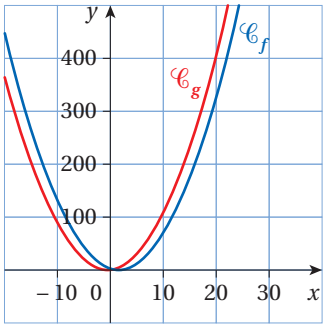
COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Adapter un algorithme aux contraintes du langage de programmation

NOTION

- Instruction conditionnelle

Deux courbes $\mathscr{C}_f$ et $\mathscr{C}_g$ sont tracées dans le repère ci-contre. Elles correspondent aux deux fonctions f et g définies sur $\mathbb{R}$ par $f(x) = 1{,}001x^2 - 3{,}026x + 1$ et $g(x) = 0{,}999x^2 + x + 1$.



a. Programmer l'algorithme suivant et expliquer ce que signifie le nombre affiché.

```
x prend la valeur 0
Tant que f(x) < g(x)
    x prend la valeur x + 1
Fin tant que
Afficher x
```

```
position_relative():={
  x:=1;
  tantque 1.001*x^2-3.026*x+1<0.999*x^2+x+1 faire
    x:=x+1;
  ftantque
  afficher("x = ",x);
};;
```

```
position_relative()
"x = ",2013
```

Le nombre affiché, 2013, est le plus petit entier n supérieur à 0 tel que $f(n-1) < g(n-1)$.
$f(2012) < g(2012)$, $f(2013) = g(2013)$ et $f(2014) > g(2014)$.

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

b. Modifier l'algorithme précédent pour qu'il donne pour l'ensemble des entiers n compris entre 0 et 2013, la valeur la plus grande de la différence $f(n) - g(n)$.

```
d prend la valeur 0
Pour n allant de 0 à 2013
    Si g(n) - f(n) > d alors
        d prend la valeur g(n) - f(n)
        n prend la valeur n + 1
    Fin si
Fin pour
Afficher d et n
```

La valeur la plus grande de $f(n) - g(n)$ est 2 026,08 et est atteinte pour $n = 1\,006$ et pour $n = 1\,007$.

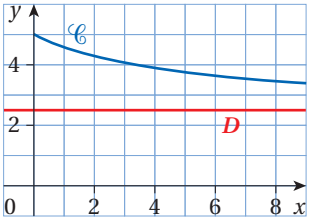
24

Prérequis : Position relative de courbes

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS
- Affectation, variable
- Boucle conditionnelle

Soit f la fonction définie sur l'intervalle $[0 ; +\infty[$ par $f(x) = \dfrac{5x+50}{2x+10}$, $\mathscr{C}$ sa courbe représentative dans un repère orthogonal du plan et D la droite d'équation $y = 2{,}5$.



Un logiciel de calcul formel donne des informations sur la différence $\dfrac{5x+50}{2x+10} - 2{,}5$:

$\dfrac{5\cdot x+50}{2\cdot x+10}-2.5$ → $\dfrac{25}{2\cdot(x+5)}$

$\text{solve}\left(\dfrac{25}{2\cdot(x+5)}<\dfrac{1}{1000},x\right)$ → $x<-5$ or $x>12495$

a. On donne l'algorithme suivant :

```
x prend la valeur 0
d prend la valeur 0
Tant que d > 0,001
    d prend la valeur (5x + 50)/(2x + 10) − 2,5
    x prend la valeur x + 1
Fin tant que
Afficher x
```

Programmer cet algorithme sur la calculatrice et constater que la valeur affichée est 0.

```
PROGRAM:PROCHE
:0→X:0→D
:While D>0.001
:(5X+50)/(2X+10)
-2.5→D
:X+1→X
:End
:Disp X
```

```
prgmPROCHE
                0
Done
```

b. Indiquer comment modifier l'algorithme pour qu'il affiche 12 496.

Il faut prendre comme valeur initiale pour d un nombre supérieur à 0,001 pour que la boucle conditionnelle soit parcourue une première fois. Pour d = 1, le programme à la calculatrice donne comme affichage 12 496, mais le temps de calcul est très long.

Équations et inéquations

25

Prérequis : Résolution d'équations

COMPÉTENCES ALGORITHMIQUES
- Modifier un algorithme
- Mettre au point une solution algorithmique

NOTIONS
- Affectation, variable
- Instruction conditionnelle

On souhaite résoudre une équation du type $ax^2 = b$ où a et b sont des nombres réels donnés.

1 Compléter la 3e et la 5e ligne de l'algorithme ci-après écrit en langage naturel.

```
Entrer les réels a et b
Si ab < 0 alors
    Afficher « L'équation n'admet pas de solution réelle »
Sinon
    Afficher « L'équation admet deux solutions : x = √(b/a) ou x = −√(b/a) »
Fin si
```

2 **a.** Tester ce programme avec les valeurs $a = 0$ et $b = 3$. Que constate-t-on ? Expliquer.

Le programme renvoie une erreur mathématique car il mène à une division par 0.

b. Modifier le programme de manière à pouvoir traiter les situations où le coefficient a est nul.

De nombreuses solutions sont possibles. En voici l'une d'entre elles :

```
Entrer les réels a et b
Si a ≠ 0 alors
    Si ab < 0 alors
        Afficher « L'équation n'admet pas de solution réelle »
    Sinon
        Afficher « L'équation admet deux solutions : x = √(b/a) ou x = −√(b/a) »
    Fin si
Sinon
    Si b = 0
        Afficher « Tout réel x est solution »
    Sinon
        Afficher « L'équation n'admet pas de solution »
    Fin si
Fin si
```

26

Prérequis : Résolution d'équations

COMPÉTENCES ALGORITHMIQUES

- Identifier les données d'entrée, de sortie, le traitement
- Mettre au point une solution algorithmique

NOTIONS

- Affectation, variable
- Instruction conditionnelle

On veut pouvoir vérifier automatiquement si des valeurs données sont solutions d'une équation quelconque de la forme $A(x) = B(x)$, où $A(x)$ et $B(x)$ sont des expressions algébriques.

1 Effectuer cette vérification avec $A(x) = 2x^2 - x - 13$ et $B(x) = x - \dfrac{3}{x}$ avec les valeurs suivantes :

a. $a = 2$.

Pour $a = 2$, $A(2) = 2 \times 2^2 - 2 - 13 = -7$ et $B(2) = 2 - \dfrac{3}{2} = 0{,}5 \neq -7$.

2 n'est pas solution de l'équation $A(x) = B(x)$.

b. $a = 3$.

Pour $a = 3$, $A(3) = 2 \times 3^2 - 3 - 13 = 2$ et $B(3) = 3 - \dfrac{3}{3} = 2$.

3 est solution de l'équation $A(x) = B(x)$.

2 On veut créer un algorithme permettant de tester si une valeur est solution d'une équation quelconque de la forme $A(x) = B(x)$.

a. Quelles sont les trois entrées nécessaires ?

$A(x)$, $B(x)$ et x.

b. Quels sont les traitements possibles parmi :

- [x] Effectuer des calculs numériques.
- [ ] Effectuer une boucle pour réitérer une action.
- [x] Effectuer un test.
- [x] Mettre en œuvre une instruction conditionnelle.

c. Quelles sont les sorties possibles ?

« x est solution de l'équation $A(x) = B(x)$. »

« x n'est pas solution de l'équation $A(x) = B(x)$. »

d. Écrire cet algorithme en langage naturel.

```
Entrer le réel positif x, les expressions A et B
Calculer A(x) et B(x)
Si A(x) = B(x) alors
    Afficher « x est solution »
Sinon
    Afficher « x n'est pas solution »
Fin si
```

27

Prérequis : Résolution d'équations et d'inéquations

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Identifier les données d'entrée, de sortie, le traitement
- Valider la solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

- Affectation, variable
- Instruction conditionnelle
- Boucle conditionnelle

Au début du XVIII^{e} siècle, un marchand veut remonter de Sète jusqu'à Toulouse pour vendre sa farine. Pour cela, il emprunte le canal du Midi qui relie la mer Méditerranée et la Garonne. Ce canal est parsemé de 63 écluses. À chacune d'elles, le marchand doit laisser 1 % de son chargement en péage royal, puis échanger 5 sacs de farine contre de la nourriture.

L'objectif est de déterminer la quantité de farine qu'il lui reste à vendre à son arrivée à Toulouse.

```
>>> def nbsacs(a):
        i=63
        while i>0:
            a=a*0.99-5
            i=i-1
        return a

>>> nbsacs(7000)
3481.7915721633512
>>> nbsacs(10000)
5074.5082010286887
>>> nbsacs(5000)
2419.9804862531241
>>>
```

L'algorithme ci-contre, écrit en Python, calcule le nombre de sacs restants à Toulouse en fonction du nombre a de sacs que le marchand avait au départ de Sète. On a ensuite effectué quelques tests pour différentes valeurs de a.

1 Expliquer la ligne « a=a*0.99–5 ».

À chaque écluse, il laisse 1 % de son chargement et 5 sacs de farine, il lui reste donc $a \times 0{,}99 - 5$.

L'égalité « a=a*0.99–5 » donne la nouvelle affectation de a.

2 Que représente i ?

i représente le nombre d'écluses restant à franchir.

3 Pour $a = 10\,000$, compléter le tableau des premières étapes de l'algorithme.

	Départ	1re écluse	2e écluse	3e écluse	4e écluse	5e écluse
a	10 000	9 895	9 791,05	9 688,14	9 586,26	9 485,4
i	63	62	61	60	59	58

4 Estimer le nombre d'opérations (soustractions, multiplications) réalisées lorsque s'exécute la fonction **nbsacs**.

Chaque boucle comporte une multiplication et deux soustractions, soit au total 63 multiplications et 126 soustractions.

Vecteurs et repères

28

Prérequis : Coordonnées d'un vecteur, norme d'un vecteur

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme

NOTION
- Affectation, variable

Vanessa a programmé sur sa calculatrice le calcul des coordonnées $(x\,;\,y)$ d'un vecteur $\overrightarrow{AB}$ à partir de deux points repérés $A(m\,;\,n)$ et $B(p\,;\,q)$.

```
PROGRAM:COORDVEC
:Prompt M,N,P,Q
:P-M→X
:Q-N→Y
:Disp X,Y
```

Cynthia lui fait remarquer qu'avec une ligne supplémentaire, elle obtiendrait aussi la norme du vecteur $\overrightarrow{AB}$.

a. Modifier le programme de Vanessa comme le suggère Cynthia.

On peut ajouter une ligne de la forme : SQRT($X^2 + Y^2$) → Z, et ajouter Z dans la liste des résultats à afficher en sortie.

b. Calculer les coordonnées et la norme du vecteur $\overrightarrow{AB}$ pour $A(4\,;\,-8)$ et $B(-1\,;\,2)$.

On trouve : $\overrightarrow{AB}(-5\,;\,10)$ et $||\overrightarrow{AB}|| = 5\sqrt{5}$.

29

Prérequis : Colinéarité de vecteurs, alignement de trois points

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Valider la solution algorithmique

NOTIONS
- Affectation, variable
- Instruction conditionnelle

Voici ci-contre un algorithme programmé en Python qui teste la colinéarité de deux vecteurs.
Malheureusement, une erreur s'est glissée dans le programme.

```
def colinearite():
    xa=input ('abscisse de A');
    ya=input ('ordonnee de A');
    xb=input ('abscisse de B');
    yb=input ('ordonnee de B');
    xc=input ('abscisse de C');
    yc=input ('ordonnee de C');
    xd=input ('abscisse de D');
    yd=input ('ordonnee de D');
    m=xb-xa;
    n=yb-ya;
    p=xd-xc;
    q=yd-yc;
    if m*p-n*q==0:
        print 'les vecteurs sont colineaires'
    else:
        print 'les vecteurs ne sont pas colineaires'
```

a. Retrouver et corriger l'erreur.

L'erreur vient du test de colinéarité qui devrait être : « if m*q – n*p == 0 ».

b. Adapter ce programme pour tester l'alignement de trois points.

Il faut trois points A, B et C et remplacer le calcul des coordonnées du vecteur $\overrightarrow{CD}$ par celui des coordonnées du vecteur $\overrightarrow{AC}$ par exemple. Penser bien sûr à changer le texte dans la conclusion.

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

```
def colinearite():
    xa=input ('abscisse de A');
    ya=input ('ordonnee de A');
    xb=input ('abscisse de B');
    yb=input ('ordonnee de B');
    xc=input ('abscisse de C');
    yc=input ('ordonnee de C');
    m=xb-xa;
    n=yb-ya;
    p=xc-xa;
    q=yc-ya;
    if m*q-n*p==0:
        print 'les points sont alignés'
    else:
        print 'les points ne sont pas alignés'
```

c. Les points $A(3\,;\,-1)$, $B(-1\,;\,2)$ et $C(-4\,;\,5)$ sont-ils alignés ?

Les points A, B et C ne sont pas alignés.

30

Prérequis : Distance dans un repère orthonormé

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Valider la solution algorithmique

NOTIONS

- Affectation, variable
- Instruction conditionnelle

Pour tester si un triangle ABC est isocèle à partir de ses sommets repérés, on construit l'algorithme suivant :

```
Entrer xa et ya les coordonnées de A
Entrer xb et yb les coordonnées de B
Entrer xc et yc les coordonnées de C
a prend la valeur (xb − xc)² + (yb − yc)²
b prend la valeur (xa − xc)² + (ya − yc)²
Si a = b alors
    Afficher « Le triangle ABC est isocèle en C »
Sinon
    c prend la valeur (xa − xb)² + (ya − yb)²
    Si a = c alors
        Afficher « Le triangle ABC est isocèle en B »
    Sinon
        Si b = c alors
            Afficher « Le triangle ABC est isocèle en A »
        Sinon
            Afficher « Le triangle ABC n'est pas isocèle »
        Fin si
    Fin si
Fin si
```

1 Tester l'algorithme avec les points A(−1 ; −1), B(7 ; 3) et C(2 ; 3). Qu'obtient-on en sortie ?

Le test « Si $a = b$ » est vrai, donc l'algorithme affiche « Le triangle ABC est isocèle en C ».

2 **a.** Que représentent les variables a, b et c ?

Les variables a, b et c représentent les carrés des longueurs BC, AC et AB.

b. Pourquoi le calcul de c n'est-il pas réalisé en même temps que celui de a et b ?

Le calcul de c n'est nécessaire que si a et b ne sont pas égaux.

3 **a.** Modifier l'algorithme afin qu'il affiche plutôt « Le triangle ABC est équilatéral » lorsque c'est le cas.

```
Entrer xa et ya les coordonnées de A
Entrer xb et yb les coordonnées de B
Entrer xc et yc les coordonnées de C
a prend la valeur (xb − xc)² + (yb − yc)²
b prend la valeur (xa − xc)² + (ya − yc)²
c prend la valeur (xa − xb)² + (ya − yb)²
Si a = b alors
    Si a = c alors
        Afficher « Le triangle ABC est équilatéral »
    Sinon
        Afficher « Le triangle ABC est isocèle en C »
    Fin si
Sinon
    Si b = c alors
        Afficher « Le triangle ABC est isocèle en A »
    Sinon
        Si a = c alors
            Afficher « Le triangle ABC est isocèle en B »
        Sinon
            Afficher « Le triangle ABC n'est pas isocèle »
        Fin si
    Fin si
Fin si
```

b. Tester l'algorithme avec les points A(2 ; 3), B(–2 ; 0) et C(2,5 ; –2). Qu'obtient-on en sortie ?

On obtient en sortie « Le triangle ABC n'est pas isocèle ».

31

Prérequis : Parallélogramme, angles

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Valider la solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

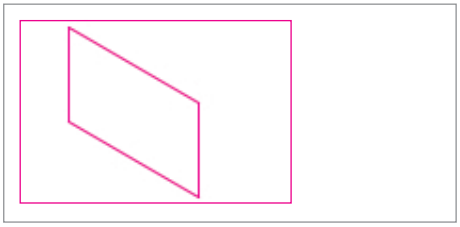
- Affectation, variable
- Boucle itérative
- Boucle conditionnelle

Dans cet exercice, on utilise le logiciel GéoTortue (les fonctions et commandes de ce module sont expliquées en introduction de ce cahier p. 9).

1 La procédure **truc** est définie par les instructions ci-dessous.

```
1 > pour truc a l L
2 > av l
3 > tg a
4 > av L
5 > tg 180-a
6 > av l
7 > tg a
8 > av L
```

a. Construire **truc 60 50 80**.



b. Justifier la nature de l'objet réalisé par la procédure **truc**.

On peut démontrer par exemple que les côtés opposés ont la même longueur et que les angles opposés ont la même mesure.

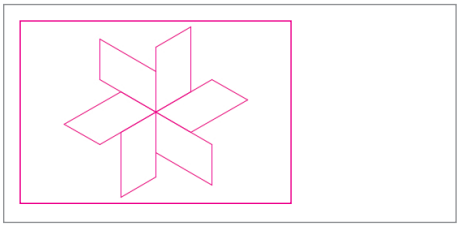
2 La procédure **multitruc** est définie par les instructions ci-dessous.

```
1 > pour multitruc n
2 > a:=360/n
3 > i:=1
4 > tant_que (i<=n) [ truc a 50 80 ; tg 180 ; i:=i+1 ]
```

a. Si $n = 4$, quelle est la nature des figures qui composent **multitruc 4** ?

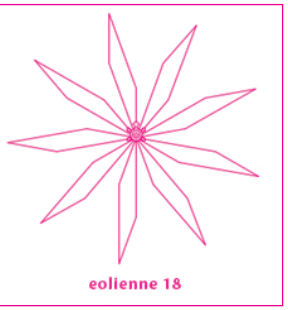
Il s'agit de 4 rectangles semblables.

b. Construire **multitruc 6**.



3 Définir la procédure **eolienne**, qui construit une figure constituée d'une répétition de n parallélogrammes de dimensions 50 et 80, n étant variable. Une **eolienne** est un **multitruc** dont seul un parallélogramme sur deux est dessiné.

```
1 > pour eolienne n
2 > a:=360/n
3 > i:=1
4 > tant_que (i<n/2) [ truc a 50 80 ; tg 180 ; i:=i+1 ; tg a]
5 >
6 > pour truc a l L
7 > av l
8 > tg a
9 > av L
10 > tg 180-a
11 > av l
12 > tg a
13 > av L
```



Équations de droites

32

Prérequis : Équations de droites

COMPÉTENCE ALGORITHMIQUE
- Comprendre et analyser un algorithme préexistant

NOTIONS
- Affectation, variable
- Boucle itérative

Que réalise la fonction **mystere** définie dans le programme en Python ci-contre ?

```
>>> def mystere(a,b):
        x=0
        y=b
        for i in range(10):
            print '(',x,';',y,')'
            x=x+1
            y=y+a
```

La fonction **mystere** permet d'afficher les coordonnées de points de la droite d'équation $y = ax + b$ pour des valeurs entières successives de x.

33

Prérequis : Équations de droites (pente, coefficient directeur)

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Identifier les données d'entrée, de sortie, le traitement
- Valider la solution algorithmique

NOTIONS
- Affectation, variable
- Boucle itérative

Jessica a programmé en Python un algorithme qui permet de déterminer l'équation d'une droite passant par deux points donnés.

```
>>> def droite(xa,ya,xb,yb):
        a=(yb-ya)/(xb-xa)
        b=yb-a*xb
        print 'y=',a,'x+',b

>>> droite(2,5,4,9)
y= 2 x+ 1
```

1 La fenêtre ci-contre montre ce programme ainsi que l'essai qu'elle a effectué.

a. Avec quels points A et B a-t-elle testé son programme ?

Avec le point A(2 ; 5) et le point B(4 ; 9).

b. Vérifier le résultat proposé par le programme.

On calcule la pente de la droite (AB) : $a = \frac{y_B - y_A}{x_B - x_A} = \frac{9-5}{4-2} = 2$.

La droite (AB) a donc une équation de la forme $y = 2x + b$.

Le point A appartient à la droite (AB), donc $y_A = 2x_A + b$, soit $5 = 2 \times 2 + b$, d'où $b = 1$.

L'équation de la droite (AB) est $y = 2x + 1$.

2 Jessica prête son programme à Azim qui cherche une équation de droite. Il saisit **droite(4,5,4,7)** mais le programme lui retourne une erreur. Expliquer ce phénomène.

Ici $x_B - x_A = 4 - 4 = 0$, donc le calcul du coefficient directeur est impossible (division par 0).

Ceci est dû au fait que les points A et B ont la même abscisse pour une ordonnée différente, et donc la droite n'admet pas de pente.

3 Modifier le programme de Jessica afin qu'il tienne compte de la situation rencontrée par Azim.

```
>>> def droite(xa,ya,xb,yb):
    if xa==xb:
        print 'x=',xa
    else:
        a=(yb-ya)/(xb-xa)
        b=yb-a*xb
        print 'y=',a,'x +',b


>>> droite(4,5,4,7)
x= 4
```

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

34

Prérequis : Équations de droites, positions relatives de deux droites

COMPÉTENCES ALGORITHMIQUES

- Modifier un algorithme
- Mettre au point une solution algorithmique

NOTIONS

- Affectation, variable
- Instruction conditionnelle

On souhaite trouver, s'il existe, le point d'intersection des deux droites d'équations $y = ax + b$ et $y = mx + p$, avec a, b, m et p des réels donnés.

1 En supposant que ces deux droites sont sécantes, donner la formule permettant de calculer x en fonction de a, b, m et p.

Un couple $(x ; y)$, vérifiant les équations $y = ax + b$ et $y = mx + p$, est tel que $ax + b = mx + p$, d'où $(a - m)x = p - b$. En conclusion, $x = \dfrac{p-b}{a-m}$.

2 Tester la séquence ci-dessous, écrite en Python, pour les droites d'équations $y = x + 2$ et $y = 3x + 4$.

```
>>> def inter(a,b,m,p):
        x=(p-b)/(a-m)
        y=a*x+b
        print 'il y a un couple solution: (',x,';',y,')'
```

Donner le résultat affiché en sortie.

Le programme affiche « il y a un couple solution : (–1 ; 1) ».

3 Le programme précédent n'envisage que le cas où il existe un unique point d'intersection.

a. À quelle condition les deux droites sont-elles parallèles ?

Les deux droites sont parallèles si et seulement si $a = m$.

b. À quelle condition les deux droites sont-elles confondues ?

Les deux droites sont confondues si et seulement si $a = m$ et $b = p$.

c. Écrire en langage naturel l'algorithme permettant de traiter les trois cas envisageables.

```
Entrer les nombres a, b, m et p
Si a = m alors
    Si b = p alors
        Afficher « Les droites sont confondues »
    Sinon
        Afficher « Les droites sont strictement parallèles »
    Fin si
Sinon
    x prend la valeur (p − b)/(a − m)
    y prend la valeur ax + b
    Afficher « Les droites sont sécantes au point de coordonnées (x ; y) »
Fin si
```

4 **a.** Programmer en Python l'algorithme de la question **3.c.**

```
def inter(a,b,m,p):
    if a==m:
        if b==p:
            print('les droites sont confondues')
        else:
            print('les droites sont strictement parallèles')
    else:
        x=(p-b)/(a-m)
        y=a*x+b
        print('les droites sont sécantes au point de coordonnées (',x,';',y,')')
```

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

b. Tester ce programme avec les droites d'équations $y = -5x + 4$ et $y = -5x - 3$. Donner le résultat affiché.

Le résultat affiché en sortie est : « Les droites sont strictement parallèles ».

Trigonométrie et triangle rectangle

35

Prérequis : Trigonométrie dans le triangle rectangle

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Valider la solution algorithmique

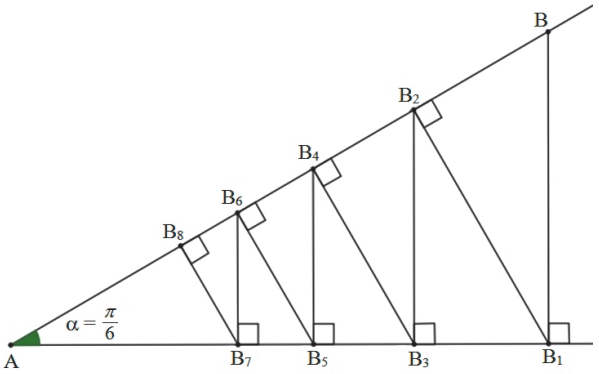
NOTIONS

- Affectation, variable
- Boucle conditionnelle

On étudie la trajectoire d'un mobile partant du point B selon le chemin indiqué sur la figure ci-contre.

Les angles en les points B_1, B_2, B_3, … , B_8 sont droits.

Le but de l'exercice est d'estimer la longueur du trajet $BB_1 + B_1B_2 + \ldots + B_{n-1}B_n$ pour différentes valeurs de l'entier naturel n, nombre de segments de la ligne brisée. On prend AB comme longueur unité.



1 **a.** Calculer BB_1.

Dans le triangle ABB_1, on a immédiatement $BB_1 = \sin\frac{\pi}{6}$.

b. Exprimer B_1B_2 en fonction de BB_1 puis, de façon générale, la longueur d'un segment du trajet en fonction de la longueur du segment précédent de la ligne brisée.

$B_1B_2 = BB_1 \times \cos\frac{\pi}{6}$. De façon générale $B_{k+1}B_k = B_kB_{k-1} \times \cos\frac{\pi}{6}$ pour tout entier $k \geqslant 2$.

c. On souhaite écrire un algorithme qui calcule une valeur approchée du trajet. Compléter le bloc de la boucle conditionnelle dans le programme écrit en Python ci-dessous.

```
from math import *
n=int(input("n=?"))
i=1
L=sin(pi/6)
trajet=L
while i<=n:
    i = i + 1
    trajet=L+trajet*cos(pi/6)
print(trajet)
```

d. Faire tourner ce programme pour $n = 10$, $n = 100$, $n = 10000$. Que constate-t-on ?

Pour $n = 10$: trajet ≈ 2,965 ; pour $n = 100$: trajet ≈ 3,732 ; pour $n = 10000$: trajet ≈ 3,732058.

On constate que la longueur du trajet « tend » vers une valeur finie, dont une valeur approchée est 3,732058.

2 **Cas général** : écrire un programme qui calcule une valeur approchée du trajet identique au précédent dans le cas où une mesure de l'angle $\widehat{BAB_1}$ et la longueur AB sont entrées par l'utilisateur.

```
from math import *
n=int(input("n=?"))
angle=eval(input("alpha=?"))
a=eval(input("longueur AB ?"))
i=1
L=a*sin(angle)
trajet=L
while i<=n:
    i=i+1
    trajet=L+a*trajet*cos(angle)
print(trajet)
```

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

36

Prérequis : Théorème de Pythagore

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Identifier les données d'entrée, de sortie, le traitement
- Mettre au point une solution algorithmique
- Valider la solution algorithmique

NOTIONS

- Affectation, variable
- Instruction conditionnelle

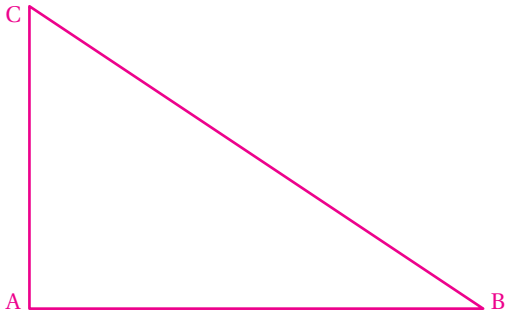
Dans un triangle rectangle ABC, la donnée des longueurs de deux de ses côtés permet de calculer la longueur du troisième côté.

1 Dans les deux cas ci-dessous, faire une figure et calculer la longueur BC.

a. On sait que les segments [AB] et [AC] sont les côtés de l'angle droit et on donne AB = 6 cm et AC = 4 cm.

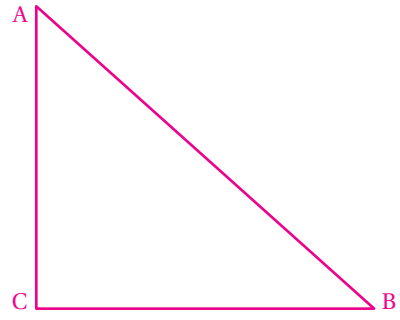
b. On sait que l'hypoténuse est connue et on donne AB = 6 cm et AC = 4 cm.

a.



$BC = \sqrt{AB^2 + AC^2} = \sqrt{6^2 + 4^2} = \sqrt{52} = 2\sqrt{13}$

b.



$BC = \sqrt{AB^2 - AC^2} = \sqrt{6^2 - 4^2} = \sqrt{20} = 2\sqrt{5}$

2 **Étude d'un algorithme**

Pour automatiser cette tâche, on propose l'algorithme ci-contre.

```
PROGRAM:TRIRECT
:Input "HYP CONN
UE : 0 OU 1",H
:Input "AB",X
:Input "AC",Y
:If H=0
:Then
:√(X^2+Y^2)→Z
:Else
:√(X^2-Y^2)→Z
:End
:Disp "BC",Z
```

a. Exécuter cet algorithme dans les deux cas vus précédemment, et remplir le tableau ci-dessous.

	H	X	Y	Z
Cas n° 1	0	6	4	$2\sqrt{13}$
Cas n° 2	1	6	4	$2\sqrt{5}$

b. Que donne cet algorithme si, en entrée, on annonce que l'hypoténuse est connue et que AB = 4 cm et AC = 6 cm ?

L'hypoténuse est alors AC, mais avec $H = 1$, l'algorithme calculera $\sqrt{AB^2 - AC^2}$ qui n'existe pas car AB < AC.

c. Réécrire l'algorithme en langage naturel pour qu'il soit fonctionnel dans toutes les situations.

```
Afficher « Hypoténuse connue : 0 ou 1 ? », H prend la valeur de la réponse
Afficher « AB », X prend la valeur de la réponse
Afficher « AC », Y prend la valeur de la réponse
Si H = 0 alors
    Z prend la valeur √(X² + Y²)
Sinon
    Si X > Y alors
        Z prend la valeur √(X² − Y²)
    Sinon
        Z prend la valeur √(Y² − x²)
    Fin si
Fin si
Afficher Z
```

Remarque : le cas où AB = AC fonctionnera aussi en retournant $Z = 0$, le cas d'un triangle où B et C sont confondus.

37

Prérequis : Langage GéoTortue (voir p. 9)

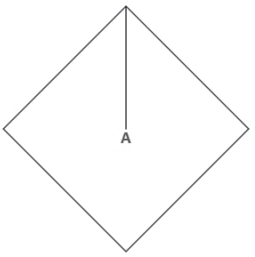
COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Valider la solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

- Affectation, variable
- Boucle itérative

On souhaite dessiner la figure ci-contre à l'aide d'un algorithme, programmé avec GéoTortue. Le point de départ sera A.

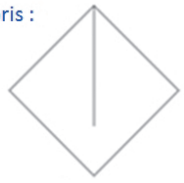


Alex propose l'algorithme ci-dessous, écrit avec GéoTortue.

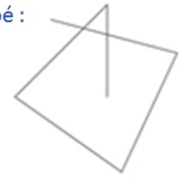
```
Fenêtre de commande
>av 50*sqrt(2)
>tg 135
>rep 4 [av 100; tg 90]
```

Boris, Chloé et Donan, qui ont commis une erreur de programmation, obtiennent les figures suivantes :

Boris :



Chloé :



Donan :

a. Écrire, dans le même langage de programmation, les programmes erronés des trois élèves.

Boris :

```
Fenêtre de commande
>av 100
>tg 135
>rep 4 [av 100; tg 90]
```

Chloé :

```
Fenêtre de commande
>av 50*sqrt(2)
>tg 135
>rep 4 [av 100; tg 100]
```

Donan :

```
Fenêtre de commande
>av 50*sqrt(2)
>tg 45
>rep 4 [av 100; tg 90]
```

b. Ajouter des instructions à l'algorithme d'Alex pour qu'il revienne au point A une fois la figure réalisée.

On ajoute les instructions : `> tg 45`

`> av 50*sqrt(2)`

Géométrie dans l'espace

38

Prérequis : Solides de l'espace

COMPÉTENCES ALGORITHMIQUES

- Identifier les données d'entrée, de sortie, le traitement
- Mettre au point une solution algorithmique
- Valider la solution algorithmique

NOTION

- Affectation, variable

Euler a établi que tous les solides convexes* respectaient la formule suivante :

$S + F = A + 2$

où A est le nombre d'arêtes,
S est le nombre de sommets,
F est le nombre de faces.

On souhaite programmer un algorithme qui calcule le nombre d'arêtes d'un solide convexe.

* On appelle **solide convexe** un solide tel que tout segment reliant deux points intérieurs au solide reste entièrement à l'intérieur du solide.

1 **a.** Quelles sont les données nécessaires à fournir en entrée ? S et F.

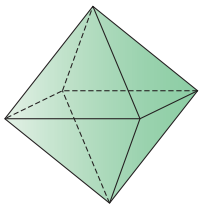
b. Quel calcul doit effectuer l'algorithme ? $A = S + F - 2$

2 Écrire l'algorithme en langage naturel.

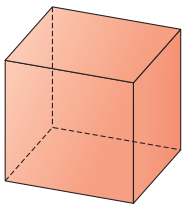
```
Entrer deux entiers naturels S et F
A prend la valeur S + F − 2
Afficher A
```

3 Calculer à l'aide de l'algorithme le nombre d'arêtes des solides convexes suivants, puis vérifier ces résultats sur les figures.

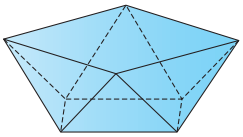
a.



b.



c.



a. 6 sommets et 8 faces. $A = S + F - 2 = 6 + 8 - 2 = 12$ arêtes.

b. 8 sommets et 6 faces. $A = S + F - 2 = 8 + 6 - 2 = 12$ arêtes.

c. 8 sommets et 10 faces. $A = S + F - 2 = 8 + 10 - 2 = 16$ arêtes.

4 Le ballon de football n'est pas rond : il est composé de 12 pentagones et 20 hexagones réguliers. Il possède 60 sommets. Pour coudre à la main ce ballon, un ouvrier a besoin d'une minute par couture (c'est-à-dire par côté).

Combien de temps est nécessaire pour coudre ce ballon ?

$S = 60$ et $F = 12 + 20 = 32$, donc $A = S + F - 2 = 60 + 32 - 2 = 90$.

Il y a 90 arêtes donc il faut 90 minutes pour coudre ce ballon à la main.

39

Prérequis : Pourcentages

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Identifier les données d'entrée, de sortie, le traitement
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

- Affectation, variable
- Boucle conditionnelle

La pyramide de Khéops est construite avec des pavés pesant 2,5 tonnes chacun. On estime que les zones creuses de la pyramide représentent 20 % de l'ensemble. On veut déterminer la masse de cette pyramide.

La pyramide est à base carrée. On dénombre 212 pavés sur le côté du premier étage, 211 pavés sur le côté du deuxième étage, et ainsi de suite, à chaque nouvel étage, un pavé de moins par côté qu'à l'étage précédent.

1 Voici un algorithme de calcul de la masse de la pyramide en tonnes.

```
N prend la valeur 212
S prend la valeur 0
Tant que N > 0
    | S prend la valeur S + N²
    | N prend la valeur N − 1
Fin tant que
M prend la valeur 0,8 × 2,5 × S
Afficher M
```

a. Compléter le tableau suivant des valeurs successives de N et S dans l'algorithme :

S	0	44 944	89 465	133 565	177 246	220 510
N	212	211	210	209	208	207

b. Que représente la valeur de S lorsque l'algorithme sort de la boucle ?

S représente le nombre total de pavés de la pyramide pleine.

c. Expliquer le calcul « $M = 0{,}8 \times 2{,}5 \times S$ ». Que représente la valeur de M ?

On multiplie S par 2,5 pour obtenir la masse de la pyramide car chaque pavé pèse 2,5 tonnes.

On multiplie le résultat par 0,8 car 20 % de la pyramide est creuse. Diminuer de 20 % revient à multiplier par 1 – 20 % = 1 – 0,2 = 0,8. M représente la masse totale de la pyramide.

2 Programmer cet algorithme sur calculatrice et déterminer la masse de la pyramide de Khéops.

Sur TI-83 + :

```
PROGRAM:KHEOPS
:212→N
:0→S
:While N>0
:S+N²→S
:N-1→N
:End
:0.8*2.5*S→M
:Disp M
```

Le programme donne une masse de 6 397 100 tonnes.

Remarque En réalité, la pyramide de Khéops a une masse estimée à 5 000 000 de tonnes, car elle repose sur une petite butte qui a permis d'économiser bon nombre de blocs de pierre !

40

Prérequis : Les volumes des solides usuels

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Identifier les données d'entrée, de sortie, le traitement

NOTIONS

- Affectation, variable
- Instruction conditionnelle

Voici un algorithme écrit en langage naturel permettant de calculer les volumes des solides de révolution.

Aymeric a programmé cet algorithme. Il propose à Emma de tester son programme. Emma, qui veut s'amuser, propose le nombre 4 en entrée. Elle annonce : « Ton programme ne fonctionne pas : il calcule un volume même quand il ne connaît pas le solide ! ».

```
Afficher « Quel type de solide ? cône = 1,
sphère = 2, cylindre = 3 »
S prend la valeur de la réponse
Si S = 1 alors
    Demander r et h
    V prend la valeur 1/3 × π × r² × h
Sinon
    Si S = 2 alors
        Demander r
        V prend la valeur 4/3 × π × r³
    Sinon
        Demander r et h
        V prend la valeur π × r² × h
    Fin si
Fin si
Afficher V
```

a. Expliquer ce qu'a fait le programme.

Si S = 4, le programme exécute la seconde instruction « **Sinon** » car S ≠ 1 et S ≠ 2.

Il calcule donc le volume d'un cylindre.

b. Modifier le bloc d'instructions écrit en bleu dans l'algorithme afin que le programme affiche « Mauvaise saisie, recommencez » en cas d'erreur de saisie.

```
Sinon
    Si S = 3 alors
        Demander r et h
        V prend la valeur π × r² × h
    Sinon
        Afficher « Mauvaise saisie, recommencez »
    Fin si
Fin si
```

Activités algorithmiques
STATISTIQUES ET PROBABILITÉS

Statistiques

41

Prérequis : Probabilités élémentaires

COMPÉTENCES ALGORITHMIQUES
- Identifier les données d'entrée, de sortie, le traitement
- Mettre au point une solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS
- Instruction conditionnelle
- Boucle itérative

Au début de l'année 2013, des statistiques donnant l'audience de chaînes de télévision sont les suivantes :

Chaîne 1	Chaîne 2	Chaîne 3	Autres chaînes
25,5 %	17 %	11 %	46,5 %

Nous supposerons que ces chiffres restent stables dans les mois suivants.

a. On souhaite simuler le choix au hasard d'une personne parmi 1 000 téléspectateurs.
Expliquer comment ceci est réalisable à l'aide d'une urne qui contient des boules de couleurs différentes.

On peut modéliser la simulation par un tirage d'une boule dans une urne contenant 1000 boules, dont 255 rouges, 170 vertes, 110 jaunes et 465 blanches par exemple.

b. Proposer un algorithme qui permet de simuler ce choix. Programmer cet algorithme sur une calculatrice ou un logiciel.

Avec AlgoBox :

```
VARIABLES
    n EST_DU_TYPE NOMBRE
    affiche EST_DU_TYPE CHAINE
DEBUT_ALGORITHME
    n PREND_LA_VALEUR floor(1000*random()+1)
    SI (n<256) ALORS
        DEBUT_SI
        affiche PREND_LA_VALEUR "La boule est rouge"
        FIN_SI
    SI (n>=256 ET n<426) ALORS
        DEBUT_SI
        affiche PREND_LA_VALEUR "La boule est verte"
        FIN_SI
    SI (n>=426 ET n<536) ALORS
        DEBUT_SI
        affiche PREND_LA_VALEUR "La boule est jaune"
        FIN_SI
    SI (n>=536) ALORS
        DEBUT_SI
        affiche PREND_LA_VALEUR "La boule est blanche"
        FIN_SI
    AFFICHER affiche
FIN_ALGORITHME
```

c. On choisit maintenant 1000 spectateurs. Proposer un algorithme qui simule ce choix et qui renvoie le nombre de spectateurs qui ont choisi les chaînes 1, 2 et 3.

En Python :

```
from random import *

R,V,J,B=0,0,0,0
for k in range (1000):
    n=randint(1,1000)
    if n<256:
        R=R+1
    if n>=256 and n<426:
        V=V+1
    if n>=426 and n<536:
        J=J+1
    if n>=536:
        B=B+1
print (R, " spectateurs qui regardent la chaîne 1")
print (V, " spectateurs qui regardent la chaîne 2")
print (J, " spectateurs qui regardent la chaîne 3")
print (B, " spectateurs qui regardent une autre chaîne")
```

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

42

Prérequis : Paramètres d'une série statistique

COMPÉTENCES ALGORITHMIQUES
- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Adapter un algorithme aux contraintes du langage de programmation

NOTION
- Instruction conditionnelle

On considère l'algorithme suivant :

Demander un nombre N
Créer une liste **L** à N éléments (notés dans la suite L(1),…, L(N))
Demander N nombres et les affecter aux N variables L(1), L(2),…, L(N)
Ordonner par ordre croissant les termes de la liste L
Si N est impair **alors**
  MM **prend** la valeur $L\left(\frac{N+1}{2}\right)$
Sinon
  MM **prend** la valeur $\frac{L\left(\frac{N}{2}\right)+L\left(\frac{N}{2}+1\right)}{2}$
Fin si
Afficher MM

a. Que renvoie cet algorithme lorsqu'on l'applique aux nombres suivants :
- 4 pour la variable N ;
- 16 ; 12 ; 19 ; 13 pour les éléments de la liste L.

L'algorithme renvoie : L(1) = 12 ; L(2) = 13 ; L(3) = 16 ; L(4) = 19 ; la médiane MM vaut 14,5.

b. Compléter et programmer l'algorithme proposé pour qu'il affiche le premier quartile d'une liste de nombres.

```
N=int(input("Nombre de données ? "))
L=N*[0]
for i in range (N):
    L[i]=float(input("Entrer une valeur "))
## Ranger les nombres en ordre croissant
L.sort()
print(L)
## Calcul de la médiane
if N%2==0:
    MM=(L[int(N/2-1)]+L[int(N/2)])/2
else:
    MM=L[int(N/2)]
print("La médiane vaut :",MM)
## calcul du premier quartile
if N%4==0 :
    quartile1=L[int(N/4-1)]
else :
    quartile1=L[int(N/4)]
print("Le premier quartile vaut :",quartile1)
```

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

43

Prérequis : Tirage sans remise

COMPÉTENCES ALGORITHMIQUES
- Identifier les données d'entrée, de sortie, le traitement
- Mettre au point une solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS
- Instruction conditionnelle
- Boucle itérative

En statistiques, on est souvent amenés à étudier un échantillon d'une population. Notons 1, 2, 3,…, n (n est un entier naturel non nul) les individus de la population étudiée. Supposons que l'on veuille extraire s éléments de cette population ($0 < s < n$). Il faut donc tirer au hasard et sans remise s entiers de l'ensemble $\{1, 2,\ldots, n\}$. Dans la pratique, on crée une liste L de taille n, on choisit un nombre i au hasard, que l'on stocke dans la liste L et que l'on élimine des tirages suivants.

a. En notant **rand** une fonction qui renvoie un nombre aléatoire de $[0 ; 1[$, écrire une fonction qui renvoie un nombre entier aléatoire entre 1 et n.

INT(n*rand())+1, où INT désigne la partie entière.

b. Simuler un jeu de loto (choix de 6 entiers compris entre 1 et 49).

COUP DE POUCE

On pourra par exemple affecter à L(i) la valeur –1 pour exprimer que l'entier L(i) a été extrait.
On pourra ainsi effectuer un test conditionnel sur le signe de L(i).

En Scilab :

```
for i=1:49
    L(i)=i
end
for j=1:6
    i=floor(49*rand())+1
    while L(i)<0
        i=floor(49*rand())+1
    end
    L(i)=-1
    M(j)=i
end
```

REMARQUE

Le module *lycée* de Scilab contient la fonction **tirage_entier(*p*,*a*,*b*)** qui renvoie un vecteur de p entiers aléatoires compris entre a et b.

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

44

Prérequis : Tirage sans remise

COMPÉTENCES ALGORITHMIQUES

- Identifier les données d'entrée, de sortie, le traitement
- Mettre au point une solution algorithmique
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

- Instruction conditionnelle
- Boucle itérative

Abel et Berthe jouent au jeu suivant : on tire au hasard 6 nombres de l'ensemble {1, 2,…, 49}. Abel gagne la partie si le tirage contient deux entiers consécutifs, Berthe gagne dans les autres cas.

Simuler 1000 parties consécutives. Le programme renverra le nombre de parties gagnées par Abel (on pourra utiliser le programme de l'exercice précédent).

En Scilab :

```
compteur=0
for k=1:1000
    M=trier(loto(6,49))
    for j=1:5
        if M(j)+1==M(j+1) then
            compteur=compteur+1
            break
        end
    end
end
disp(compteur)
```

REMARQUE

La fonction **trier** est une fonction du module *lycée* de Scilab qui trie dans l'ordre croissant une liste de nombre. La fonction **loto(*a*,*b*)** peut s'obtenir en transformant le script de l'exercice précédent en fonction qui renvoie la liste M.

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

Probabilités

45

Prérequis : Calcul de fréquence

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

- Instruction conditionnelle
- Boucle itérative

Voici ci-dessous un algorithme programmé sur calculatrice permettant de lancer N fois un dé à 6 faces et de calculer la fréquence d'apparition du 6.

Sur TI-83+ :

```
PROGRAM:DES
:Prompt N
:0→S
:For(I,1,N)
:entAléat(1,6)→A
:If A=6
:Then
:S+1→S
:End
:End
:Disp S/N
```

Sur Casio Graph35+ :

```
======DES      ======
?→N↵
0→S↵
For 1→I To N↵
RanInt#(1,6)→A↵
If A=6↵
Then S+1→S↵
IfEnd↵
Next↵
S÷N↵
TOP BTM SRC MENU A⇔a CHAR
```

1 **a.** Écrire l'algorithme correspondant en langage naturel.

```
Entrer un entier naturel N
S prend la valeur 0
Pour i allant de 1 à N
    A prend une valeur aléatoire entière entre 1 et 6
    Si A = 6 alors
        S prend la valeur S + 1
    Fin si
Fin pour
Afficher S/N
```

b. Expliquer le rôle des variables N, S et A.

N est le nombre de lancers. S comptabilise le nombre de « 6 » réalisés. A prend à chaque lancer de dé la valeur obtenue par le dé.

c. Que représente la valeur affichée en sortie ?

La fréquence d'apparition du 6 sur N lancers.

2 La probabilité d'obtenir un 6 est de $\frac{1}{6}$. On souhaite mesurer l'écart observé lors de la simulation avec cette valeur.

a. Ajouter à l'algorithme la ligne permettant d'afficher cet écart.

`Afficher` $\frac{S}{N} - \frac{1}{6}$

b. Programmer cet algorithme.

Sur TI-83+ :

```
PROGRAM:DES
:Prompt N
:0→S
:For(I,1,N)
:entAléat(1,6)→A
:If A=6
:Then
:S+1→S
:End
:End
:Disp S/N
:Disp S/N-1/6
```

Sur Casio Graph35+ :

```
======DES      ======
?→N↵
0→S↵
For 1→I To N↵
RanInt#(1,6)→A↵
If A=6↵
Then S+1→S↵
IfEnd↵
Next↵
S÷N↵
S÷N-1÷6↵
COM CTL JUMP ?
```

46

Prérequis : Équations de droites

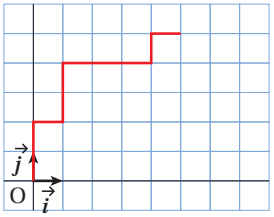
COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

- Boucle itérative
- Instruction conditionnelle

L'algorithme ci-contre, écrit sous AlgoBox, propose une marche aléatoire sur un quadrillage muni d'un repère orthonormé $(O ; \vec{i} ; \vec{j})$ d'unité 1 cm en partant de O.
On appelle « pas » un segment de longueur 1 suivant l'un des axes du quadrillage. Le résultat de son exécution est donné ci-dessous.




```
VARIABLES
    pas EST_DU_TYPE NOMBRE
    i EST_DU_TYPE NOMBRE
    A EST_DU_TYPE NOMBRE
    x EST_DU_TYPE NOMBRE
    y EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE pas
    x PREND_LA_VALEUR 0
    y PREND_LA_VALEUR 0
    POUR i ALLANT_DE 1 A pas
        DEBUT_POUR
        A PREND_LA_VALEUR ALGOBOX_ALEA_ENT(0,1)
        SI (A==1) ALORS
            DEBUT_SI
            TRACER_SEGMENT (x,y)->(x+1,y)
            x PREND_LA_VALEUR x+1
            FIN_SI
            SINON
                DEBUT_SINON
                TRACER_SEGMENT (x,y)->(x,y+1)
                y PREND_LA_VALEUR y+1
                FIN_SINON
        FIN_POUR
FIN_ALGORITHME
```

1 **a.** Quelle est la valeur de *pas* proposée en entrée ?

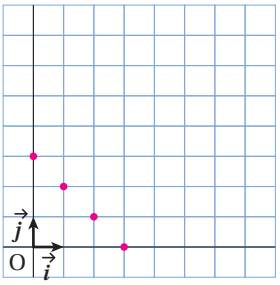
pas = 10

b. Quelle est la dernière valeur obtenue par la variable *A* ? Expliquer.

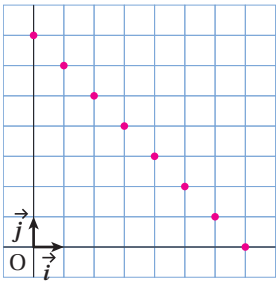
La dernière valeur obtenue pour *A* est 1 car le dernier segment est horizontal.

2 **a.** Dans les repères suivants, représenter tous les points que l'on peut atteindre en 3 pas, puis en 7 pas.

- En 3 pas :



- En 7 pas :



b. Expliquer pourquoi les points que l'on peut atteindre en *p* pas sont alignés.

Si en *p* pas l'on réalise *x* pas horizontalement et *y* pas verticalement, on a alors $x + y = p$.

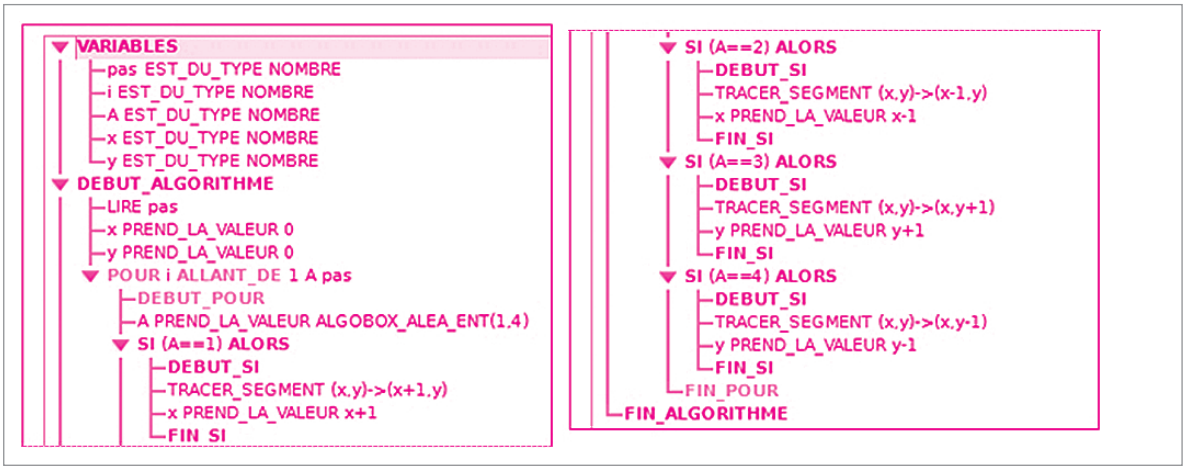
D'où $y = -x + p$; les points vérifiant cette égalité sont sur une droite.

3 Pour aller dans n'importe quelle direction sur le quadrillage, on modifie l'algorithme en choisissant un entier au hasard entre 1 et 4, et en lui associant l'une des quatre directions.

a. Écrire cet algorithme en langage naturel.

```
Entrer un entier naturel pas
x et y prennent la valeur 0
Pour I allant de 1 à pas
    | A prend une valeur aléatoire entière entre 1 et 4
    | Si A = 1 alors
    |     | Tracer le segment reliant les points de coordonnées (x ; y) et (x + 1 ; y)
    |     | x prend la valeur x + 1
    | Fin si
    | Si A = 2 alors
    |     | Tracer le segment reliant les points de coordonnées (x ; y) et (x – 1 ; y)
    |     | x prend la valeur x – 1
    | Fin si
    | Si A = 3 alors
    |     | Tracer le segment reliant les points de coordonnées (x ; y) et (x ; y + 1)
    |     | y prend la valeur y + 1
    | Fin si
    | Si A = 4 alors
    |     | Tracer le segment reliant les points de coordonnées (x ; y) et (x ; y – 1)
    |     | y prend la valeur y – 1
    | Fin si
Fin pour
```

b. Programmer cet algorithme avec AlgoBox puis exécuter-le pour 1000 pas.



47

Prérequis : Calculer une probabilité avec un arbre de probabilités

COMPÉTENCES ALGORITHMIQUES

- Comprendre et analyser un algorithme préexistant
- Modifier un algorithme
- Adapter un algorithme aux contraintes du langage de programmation

NOTIONS

- Boucle itérative
- Boucle conditionnelle
- Instruction conditionnelle

Le chevalier de Mérée (1607-1684) prétendait qu'il fallait lancer quatre fois le dé pour avoir plus de chance d'obtenir un 6 que de ne pas en obtenir.

1 Pour réaliser l'expérience, Thibaut écrit trois algorithmes et les programme.

Algorithme n° 1

```
K = 0
Pour i allant de 1 à 4
    A est un entier aléatoire
    entre 1 et 6
    Si A = 6
        K prend la valeur K + 1
    Fin si
Fin pour
Si K = 0
    Afficher « Perdu ! »
Sinon
    Afficher « Gagné ! »
Fin si
```

Algorithme n° 2

```
i = 0
Tant que i est inférieur à 4
    A est un entier aléatoire
    entre 1 et 6
    Si A = 6
        i prend la valeur 5
        Afficher « Gagné ! »
    Sinon
        i prend la valeur i + 1
    Fin si
Fin tant que
Si i = 4
    Afficher « Perdu ! »
Fin si
```

Algorithme n° 3

```
A = 0
i = 0
Tant que A est différent de 6
    A est un entier aléatoire
    entre 1 et 6
    i prend la valeur i + 1
Fin tant que
Si i est inférieur ou égal à 4
    Afficher « Gagné ! »
Sinon
    Afficher « Perdu ! »
Fin si
```

a. Expliquer le principe de chacun de ces algorithmes et de leurs variables.

- Algorithme n° 1 : lance 4 fois, compte les 6 et affiche « gagné » ou « perdu » selon le nombre de 6 obtenus.
- Algorithme n° 2 : lance au plus 4 fois, s'arrête dès que le dé vaut 6, affiche « gagné » dès que le 6 sort ou « perdu » si les 4 lancers sont effectués sans succès.
- Algorithme n° 3 : lance tant que le 6 n'est pas sorti, affiche « gagné » ou « perdu » selon le nombre de lancers effectués.

b. Thibaut a ajouté des compteurs pour déterminer le nombre de fois que la boucle de chaque algorithme s'exécute. En expérimentant chacun d'eux, il obtient comme nombre de boucles 3, 4 et 6.
À quel algorithme correspond chacun de ces résultats ?

- L'algorithme n° 1 réalise toujours 4 boucles.
- L'algorithme n° 2 réalise au plus 4 boucles, donc ici il en a réalisé 3.
- L'algorithme n° 3 a donc réalisé 6 boucles.

c. Lequel des trois algorithmes semble être le plus économe ?

L'algorithme n° 2, car il inclut les tests d'arrêts des deux autres algorithmes.

2 Thibaut exécute l'un de ses programmes de nombreuses fois mais ses résultats ne lui permettent pas de prendre parti sur l'affirmation du chevalier de Mérée. Il décide de modifier son programme pour réaliser N fois l'expérience et que ce dernier lui retourne la fréquence de parties gagnées.

a. Écrire cet algorithme en langage naturel.

```
Avec l'algorithme 2 :
Entrer un entier naturel N
G prend la valeur 0 //compteur de parties gagnées//
Pour k allant de 1 à N
    i = 0
    Tant que i est inférieur à 4
        A prend une valeur aléatoire entière entre 1 et 6
        Si A = 6
            i prend la valeur 5
            G prend la valeur G + 1
        Sinon
            i prend la valeur i + 1
        Fin si
    Fin tant que
Fin pour
Afficher G/N
```

b. Programmer cet algorithme et l'exécuter pour de grandes valeurs de N. Peut-on conclure ?

```
from random import *

N=int(input("N= ?"))
G=0
for k in range(1,N+1):
    i=0
    while i<4:
        A=randint(1,6)
        if A==6:
            i=5
            G=G+1
        else:
            i=i+1
G=G/N
print("Fréquence des parties gagnées:",G)
```

Il faut de grandes valeurs pour assurer une fréquence supérieure à 0,5.

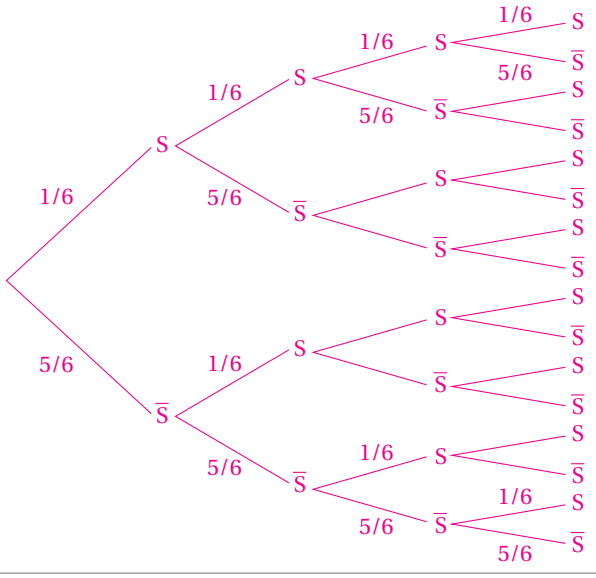
Pour N = 100 000, la fréquence se stabilise autour de 0,515.

Voir algorithme corrigé sous AlgoBox sur le site Odyssée.

3 Pour en avoir le cœur net, Thibaut construit un arbre de probabilité, en considérant qu'à chaque lancer il n'y a que deux issues : obtenir 6 ou non.

a. Construire un arbre représentant quatre lancers successifs.

En notant S l'issue « obtenir un 6 » :



b. Une seule issue est perdante. Déterminer la probabilité de perdre.

La probabilité de perdre est $\frac{5}{6}\times\frac{5}{6}\times\frac{5}{6}\times\frac{5}{6}=\frac{625}{1296}$.

c. En déduire la probabilité de gagner.

La probabilité de gagner est $1-\frac{625}{1296}=\frac{671}{1296}$.

d. Peut-on expliquer pourquoi il a fallu choisir N très grand à la question **2b** pour que l'expérience soit probante ?

$\frac{671}{1296}\approx 0{,}518$. Il a fallu un grand nombre de lancers car la probabilité de gagner est proche de 0,5.

48

Prérequis : Notion de probabilité et loi des grands nombres

COMPÉTENCE ALGORITHMIQUE
- Comprendre et analyser un algorithme préexistant

NOTIONS
- Boucle itérative
- Instruction conditionnelle

Un professeur demande à ses élèves de lancer une pièce de monnaie 50 fois et de consigner la liste des résultats « pile » (P) ou « face » (F). Voici la liste de David :

PFPPFPFFFP PFPFFPPFPP FFPPFFPPFF PFPFFFPPFP PFFFPPFPFF

1 Calculer la fréquence d'apparition du « pile ». Ce résultat semble-t-il cohérent avec l'équiprobabilité théorique de cette expérience ?

La fréquence d'apparition du pile est $\frac{23}{50}=46\,\%$. Ce résultat paraît conforme car il est assez proche de l'équiprobabilité théorique (50 %) pour un échantillon de seulement 50 expériences.

2 Le professeur n'est pas satisfait, il affirme que David a probablement triché et n'a pas lancé la pièce. David ne comprend pas comment le professeur a pu s'en rendre compte. Pour le convaincre, il propose à David de programmer l'algorithme ci-contre sur sa calculatrice.

```
PROGRAM:PILE
:0→B
:0→C
:For(I,1,50)
:entAléat(0,1)→A
:If A=1
:Then
:B+1→B
:Else
:If B>C
:Then
:B→C
:End
:0→B
:End
:End
:Disp C
```

a. Comment le programme propose-t-il de simuler un « pile » ou « face » ?

L'algorithme choisit un nombre au hasard entre 0 et 1, il suffit alors de choisir lequel représente « pile » par exemple.

b. En supposant que les quatre premières simulations donnent dans A les valeurs 1-1-1-0, quelles seront les valeurs de B et C lorsque le programme recalculera une nouvelle valeur de A ?

À chaque fois que A prend la valeur 1, B augmente de 1. B contiendra la valeur 3 après les 3 premiers lancers. Au quatrième lancer, A prend alors la valeur 0. Comme B est plus grand que C (qui vaut 0), C prend la valeur 3 et B prend la valeur 0.

c. Quel est le rôle des variables B et C dans l'algorithme ?

B compte le nombre de 1 consécutifs générés dans A. Il se réinitialise à 0 lorsqu'une série est interrompue.

C stocke la plus grande valeur de B rencontrée, c'est à dire la longueur de la plus grande série de 1 réalisée.

3 En faisant tourner le programme, voici ce que David obtient sur sa calculatrice.

```
Fait
      6
Fait
      5
Fait
     12
Fait
```

a. En regardant sa série de lancers, comprend-on la remarque du professeur ?

Dans la série de David, la plus grande série de « pile » est de longueur 2 et la plus grande série de « face » est de longueur 3. Ce résultat ne semble pas fréquent, d'où la remarque du professeur.

b. David a-t-il pu obtenir ses résultats sans tricher ?

Bien que peu probable, David aurait pu obtenir ces résultats sans tricher, par hasard.