Algorithmes en langage naturel : Programmes de terminale S et ES

Bulletin officiel spécial n°8 du 13 octobre 2011

Les activités de type algorithmique sont signalées par le symbole dans le programme de TS.

Petit listing des points du programme signalés par un symbole avec un exemple d'algorithme en langage naturel.

(lien vers algorithme sur énoncé programme officiel)

Dans le programme de ES

• Étant donné une suite (q^n) avec $0 < q < 1$, mettre en œuvre un algorithme permettant de déterminer un seuil à partir duquel q^n est inférieur à un réel a positif donné.	Exemple : Au bout de combien d'années, la population d'une ville de 10 000 habitants, qui diminue chaque année de 10% par rapport à l'année précédente, comportera-t-elle moins de 4000 habitants ? Résoudre : $q^n<a$	Donner q Donner a n prend la valeur 1 Tant que $q^n>a$ n prend la valeur n + 1 Fin Tant que Afficher n

Dans le programme de S

◇ Dans le cas d'une limite infinie, étant donnés une suite croissante (u_n) et un nombre réel A, déterminer à l'aide d'un algorithme un rang à partir duquel u_n est supérieur à A.		Donner A Donner U0 n prend la valeur 0 Tant que Un<A n prend la valeur n + 1 Fin Tant que Afficher n
Des exemples de suites récurrentes, en particulier arithmético-géométriques, sont traités en exercice. ◇ Des activités algorithmiques sont menées dans ce cadre.	Soit la suite de Fibonacci définie par : U0=U1=1; Un=U(n-1)+U(n-2) étant donne un nombre positif p écrire un algorithme calculant Un tel que Un>p	Début U0 prend la valeur 1 U1 prend la valeur 1 Donner la valeur de p i prend la valeur 2 Repeter Ui prend la valeur U0 + U1 U0 prend la valeur U1 U1 prend la valeur Ui i prend la valeur i +1 jusqua (Ui >=p) fin répéter Afficher Ui fin.

◇ Des activités algorithmiques sont réalisées dans le cadre de la recherche de solutions de l'équation $f(x)=k$.	**La dichotomie** On suppose qu'on dispose d'une fonction *f* continue et strictement croissante sur un intervalle $[a;b]$ et qui s'annule entre *a* et *b*. L'objectif est de trouver un encadrement d'amplitude maximale e (donnée) de la racine.	Saisir *a, b,e* Tant Que $b-a>e$ faire $(a+b)/2 \to c$ Si $f(a)\times f(c)<0$ Alors $c \to b$ Sinon $c \to a$ Finsi FinTant Que Afficher *a, b*
◇ Pour une fonction monotone positive, mettre en œuvre un algorithme pour déterminer un encadrement d'une intégrale.	Calculer l'intégrale d'une fonction f sur [a ; b] qui n'a pas de primitive en utilisant un algorithme. On partage [a ; b] en n sous intervalles	Début Som prend la valeur 0 C prend la valeur (b-a)/n Pour k allant de 1 à n faire Som prend la valeur som +c* f(a+k*c) Fin Pour Afficher Som Fin
◇ Des activités algorithmiques sont menées dans ce cadre, notamment pour simuler une marche aléatoire.	Un robot est placé sur la case Départ : Départ Le lancer d'une pièce bien équilibré détermine le déplacement du pion. - Pile, le robot se déplace vers la droite (on associe le réel +1) - Face, le robot se déplace vers la gauche (on associe le réel -1) - Un trajet est une succession de 4 déplacements. On veut déterminer l'événement A : « le robot est revenu à la case départ après 4 déplacements »	Debut Saisir n Mettre 0 dans G Pour j allant de 1 à n R prend la valeur 0 Pour i allant de 1 à 4 Si alea<0.5 alors R prend la valeur R+1 sinon R prend la valeur R-1 FinSi FinPour Si R=0 alors G prend la valeur G+1 FinPour Afficher G/n Fin

ALGO1

```
from math import *

q=input('q')
a=input('a')

n=1
while q**n>a:
    n=n+1

print 'Rang : ',n
```

ALGO2

```
from math import *

# suite explicite
def u(n):
    un=exp(n)
    return(un)



A=input('nombre A')

n=0
while u(n)<A:
    n=n+1

print 'Rang : ',n
```

ALGO3

```
u0=1
u1=1

p=input('p')
i=2

# python n a pas de until
while u0+u1<p:
    u=u0+u1
    u0=u1
```

```
    u1=u
    i=i+1
    print(i,u)

print(u0+u1)
print('rang',i+1)
```

ALGO4

```
from math import *
from random import *

# FONCTION DONT ON CHERCHE UNE RACINE
def f(x):

    fx=exp(x)-5

    return fx

# nombre d?riv?
def derive(f,a):
    eps=0.00001
    df=(f(a+eps)-f(a))/eps

    return(df)


#Dichotomoie racine ds [a;b] ; eps=pr?cision ; compteur = nbre de boucles
def dicho(a,b,eps):
    compteur=1
    while abs(f(a))>eps:
        c=(a+b)/2.
        if f(a)*f(c)<=0:
            b=c
        else:
            a=c
        compteur=compteur+1

    return(a,compteur)

#version recursive
def dichorec(a,b,eps,compteur):
    c=(a+b)/2.
    if abs(f(a))>eps:
        if f(a)*f(c)<=0:
            return(dichorec(a,c,eps,compteur+1))
        else:
            return(dichorec(c,b,eps,compteur+1))
```

```
    else:
        return(c,compteur)



#Newton a=d?part ; eps=pr?cision ; compteur=nombre de boucles
def newton(a,eps):
    compteur=1
    while (abs(f(a))>eps)and(derive(f,a))!=0 :
        a=-f(a)/derive(f,a)+a+0.
        compteur=compteur+1

    return(a,compteur)


#Newton a=d?part ; eps=pr?cision ; compteur=nombre de boucles
def newtonrec(a,eps,compteur):

    if (abs(f(a))>eps)and(derive(f,a))!=0:
        return(newtonrec(-f(a)/derive(f,a)+a+0.,eps,compteur+1))
    else:
        return(a,compteur)


def montecarlo(a,b,eps,maxiter):
    compteur=1
    c=randint(a,b)
    while ( abs(f(c))>eps )and( compteur<maxiter):
        c=(b-a)*random()+a
        compteur=compteur+1
    return(c,compteur)


print 'dicho:',dicho(-1,5,0.000001)
print 'dichorec:',dichorec(-1,5,0.000001,1)
print 'newtow:',newton(3,0.00000001)
print 'newtonrec:',newtonrec(3,0.00000001,1)
print 'montecarlo:',montecarlo(-1,5,0.000001,10000000)
```

ALGO5

```
def f(x):
    fx=x**2
    return(fx)


def rectanglesmin(a,b,n):
```

```
    S=0.
    h=(b-a)/n
    for i in range(0,n):
        S=S+h*f(a+h*i)
    return(S)

def rectanglesmax(a,b,n):
    S=0.
    h=(b-a)/n+0.
    for i in range(1,n+1):
        S=S+h*f(a+h*i)
    return(S)

def trapezes(a,b,n):
    S=0.
    h=(b-a)/n
    for i in range(0,n):
        t=0.5*h*(f(a+i*h)+f(a+(i+1)*h))
        S=S+t
    return(S)

print 'rectangles min:',rectanglesmin(0,1.,10000)
print 'rectangles max:',rectanglesmax(0,1.,10000)
print 'trapezes:',trapezes(0,1.,10000)
```

ALGO6

```
from random import *
N=input('nombre echantillons')
k=4
nbre=0.

for i in range(1,N+1):
    s=0
    for j in range(1,k+1):
        pf=randint(0,1)

        if pf==0:
            s=s+1
        if pf==1:
             s=s-1


    if s==0:
         nbre=nbre+1.

print('frequence :'),nbre/N
```