DS04

17 Janvier 2023

TP noté sur machine

Sources : exercice 1 : David Prévost exercice 2 : David Prévost exercice 3 : Emilien DURIF

Consignes

Vos réponses dépendent d'un paramètre α, unique pour chaque étudiant, qui vous est donné sur le site de la classe. On considère la suite u à valeurs dans $[\![0, 64\,007[\![$, définie comme suit.

$$u_0 = \alpha \text{ et } \forall n \in \mathbb{N},\ u_{n+1} = (15\,091 \times u_n)\,[64\,007].$$

Nous vous en proposons l'implémentation suivante.

```
def u(alpha,n):
    """u_n, u_0 = alpha"""
    x = alpha
    for i in range(n):
        x = (15091 * x) % 64007
    return x
```

Cette fonction sera déjà implémentée dans le notebook et est importé depuis le module `DS4_initialisation`. Vous pouvez donc directement utiliser le fonction `u(alpha,n)` avec `alpha` correspondant à votre numéro d'anonymat.

Dans ce devoir, on notera $a\%b$ le reste de la division euclidienne de a par b.

Lorsque vous donnerez un résultat flottant, vous écrirez juste ses huit premières décimales.

Vous trouverez en annexe (à la fin du sujet) les réponses pour le paramètre $\alpha = 100$, utilisez-les pour vérifier la correction de vos algorithmes.

Exercice 1 – Algorithme d'Euclide

Mise en route et initialisation

Question 1 *Pour votre valeur de alpha donner* `a=max(u(alpha,10),u(alpha,100))` *et* `b=min(u(alpha,10),u(alpha,100))`.

On donne les instructions suivantes qui permettent de compter le nombre d'appel à une fonction :

```
def compte_appel():
    global C
    C+=1


C=0
```

```
for k in range(a):
        compte_appel()
```

Question 2 *Exécuter et vérifier cette suite d'instructions puis donner la valeur de C dans votre cas qui doit correspondre à a.*

Première méthode

L'algorithme d'Euclide permet, étant donnés deux entiers a et b, de calculer leur plus grand commun diviseur (pgcd) d. Cet algorithme se base sur la propriété suivante :

$$\begin{cases} \text{pgcd}(a,b) = a \text{ si } b = 0 \\ \text{pgcd}(a,b) = \text{pgcd}(b, a \,\%\, b) \text{ sinon} \end{cases}$$

où $a \,\%\, b$ représente le reste de la division euclidienne de a par b.

Question 3 *Ecrire une fonction récursive* `pgcd(a,b)` *qui calcule le plus grand commun diviseur de deux entiers en utilisant l'algorithme d'Euclide. Donner le PGCD de* `a=max(u(alpha,10),u(alpha,100))` *et* `b=min(u(alpha,10),u(alpha,100))`.

Question 4 *Donner le nombre d'appels récursifs de la fonction* `pgcd(a,b)` *avec* `a=max(u(alpha,10),u(alpha,100))` *et* `b=min(u(alpha,10),u(alpha,100))` *en utilisant la fonction* `compte_appel` *et en ayant bien réinitialisé la variable* `C`.

Méthode utilisant le théorème de Bézout

Le théorème de Bézout nous assure également l'existence de deux entiers u et v tels que :

$$a \cdot u + b \cdot v = d$$

(u et v sont des coefficients de Bézout de a et b).

Une version étendue de l'algorithme d'Euclide permet de calculer, en plus du pgcd d des valeurs possibles pour les coefficients de Bézout u et v.

Cet algorithme prend en entrée deux entiers a et b. Il procède de la manière suivante :

- Si $b = 0$ alors $d = a$, $u = 1$ et $v = 0$.
- Sinon, on applique récursivement l'algorithme sur les entiers b et $(a \,\%\, b)$.

On obtient ainsi d', u' et v' tels que :

d' = pgcd(b, a % b) ; et : $b.u' + (a \,\%\, b).v' = d'$

On en déduit la solution pour a et b grâce aux égalités :

$d = d'$, $u = v'$ et $v = u' - (a//b).v'$

Question 5 *En déduire une fonction* `bezout(a:int,b:int)->d:int,u:int,v:int` *qui étant donnés deux entiers* a *et* b *calcule le triplet (*d,u,v*) comme expliqué ci-dessus. Donner les valeurs de* d, u *et* v *pour* `a=max(u(alpha,10),u(alpha,100))` *et* `b=min(u(alpha,10),u(alpha,100))`.

Question 6 *Donner le nombre d'appels récursifs de la fonction* `bezout(a,b)` *avec* `a=max(u(alpha,10),u(alpha,100))` *et* `b=min(u(alpha,10),u(alpha,100))` *en utilisant la fonction* `compte_appel` *dans la fonction* `bezout(a,b)` *et en ayant bien réinitialisé la variable* `C`.

Exercice 2 – PGCD rapide

On peut écrire un PGCD très rapide, uniquement à l'aide de soustractions et de divisions par deux ou de restes modulo 2 (très rapides).

L'algorithme se base sur la parité des nombres, par exemple :

- si a est pair et b pair, `pgcd(a, b) = 2 . pgcd(a/2, b/2);`
- si a est pair et b impair, `pgcd(a, b) = pgcd(a/2, b);`
- si a est impair et b pair, `pgcd(a, b) = pgcd(a, b/2);`
- si a et b sont impairs ; pgcd(a,b)=pgcd(M-m,m) avec m=min(a,b) et M=max(a,b).

En se basant sur ce type de constatations, et en traitant tous les cas possibles, on peut à chaque fois réduire le problème.

Question 7 *Ecrire la fonction qui calcule le pgcd rapide que l'on appellera* `pgcd_rapide(a:int,b:int)->int`. *Donner le nombre d'appels récursifs de cette fonction avec* `a=max(u(alpha,10),u(alpha,100))` *et* `b=min(u(alpha,10),u(alpha,100))` *en utilisant la fonction* `compte_appel` *en ayant bien réinitialisé la variable* `C`.

Exercice 3 – Représentation des nombres

Question 8 *Écrire une fonction* `binaire(e:int,n:int)->str` *qui convertit un entier* `e` *en base 2 sur* `n` *bits. Cette fonction renverra le résultat sous la forme d'une chaine de caractère sous la forme '0' ou '1' (du poids le plus faible au plus fort de la droite vers la gauche). Attention, il faut traiter les cas où le nombre obtenu est codé sur moins de* `n` *bits et complété par des 0 sur la droite. Donner alors* `a=max(u(alpha,10),u(alpha,100))` *en base 2 sur 16 bits.*

Question 9 *Donner* `a=max(u(alpha,10),u(alpha,100))` *en base 16 sous la forme d'une chaine de caractères.*

Soit le nombre flottant représenté par `a,b` donner sa représentation en virgule flottante avec la norme IEEE 754. Par exemple si `a=3` et `b=23`, on considèrerait le flottant `3.23`.

Question 10 *Donner le bit de la mantisse du flottant défini précédemment à la position numéro "alpha%52" après la virgule. (En numérotation "Python"). On pourra utiliser la fonction* `hex` *qui renvoie la représentation IEEE 754 en hexadécimal.*

Indications

Exemples de réponse pour $\alpha = 100$

Question	Réponse
Q1a	60010
Q1b	15381
Q2	60010
Q3	1
Q4	10
Q5d	1
Q5u	-2489
Q5v	9711
Q6	10
Q7	34
Q8	1110101001101010
Q9	ea6a
Q10	1