Interrogation d'ITC n°1A

Semaine du 09/10/2023

Nom :		Prénom :	

Consignes

- Les codes doivent être présentés en mentionnant explicitement l'indentation, au moyen par exemple de barres verticales sur la gauche.
- Pour qu'un code soit compréhensible, il convient de choisir des noms de variables les plus explicites possibles.

Exercice 1 **Étude de la 2-valuation d'un entier** [Sol 1] Pour tout entier $n \geq 1$, on désigne par $v(n)$ le plus grand entier k tel que 2^k divise n (c'est-à-dire tel que $\frac{n}{2^k} \in \mathbb{N}$). On a ainsi $v(2^3) = 3, v(12) = 2, v(3) = 0$.

1. On considère la fonction suivante :

```
1 def v(n:int)->int:
2     a = 1
3     k = 0
4     while n % a == 0 and a <= n:
5         k = k+1
6         a = 2*a
7     return k
```

1.1) /2 Compléter le tableau ci-après, détaillant l'évolution des variables a et k lors de l'appel v(8).

a	k
1	0
2	1
4	2
8	3
16	4

Quel est le rôle de la variable a ? Quel est le rôle de la variable k ?

a correspond aux puissances de 2 que l'on teste comme diviseur de n, et k la puissance associée. En fin de boucle, on peut observer que k vaut v(8)+1, on retourne donc plutôt k-1

1.2) /2 Recopier et modifier le programme précédent afin qu'il retourne la valeur de $v(n)$, en effectuant uniquement une modification dans la ligne 7.

```
1 def v(n:int)->int:
2     a = 1
3     k = 0
4     while n % a == 0 and a <= n:
5         k = k+1
6         a = 2*a
7     return k-1
>>> v(8)
3
>>> v(7)
0
```

2. /3 Soit N ∈ $\mathbb{N}^\star$. On souhaite trouver la valuation maximale parmi les entiers de $[\![1, N]\!]$, c'est-à-dire $\max_{n \in [\![1,N]\!]} v(n)$. Compléter le code ci-après pour qu'il retourne cet entier.

```
def max_val(N : int) -> int:
    maxi = v(1)
    for k in range(2, N+1) :
        if v(k) >= maxi:
            maxi = v(k)
    return maxi
>>> max_val(10) # valeur que l'on peut contrôler en observant \
↪ la liste de la question suivante
3
```

3. /3 On peut montrer que l'on a pour $n \geq 1$:

$$v(n) = \begin{cases} 0 & \text{si } n \text{ est impair} \\ v(\frac{n}{2}) + 1 & \text{si } n \text{ est pair.} \end{cases}$$

Sans utiliser la fonction v, écrire une fonction `listeV(n)` d'argument n un entier supérieur ou égal à 1 et qui renvoie la liste contenant les valeurs $[v(1), ..., v(n)]$.

Il faut faire ici attention au décalage entre les indices de la liste $0, \ldots, n-1$ et ceux de la suite v, qui sont $1, \ldots, n$. Il faut aussi faire attention à utiliser // pour travailler avec des indices entiers.

```python
def listeV(n:int)->list:
    L = []
    for i in range(1, n+1):
        if i % 2 == 1:
            # cas impair
            L.append(0)
        else:
            L.append(L[(i-1)//2]+1)
    return L
```

```
>>> listeV(10)
[0, 1, 0, 2, 0, 1, 0, 3, 0, 1]
>>> [v(i) for i in range(1, 11)]
[0, 1, 0, 2, 0, 1, 0, 3, 0, 1]
```

Exercice 2 **Matrices et listes de listes** [Sol 2] On rappelle un exemple d'utilisation de append dans la console.

```
>>> L = [1, 2, 3]
>>> L.append(1)
>>> L
[1, 2, 3, 1]
```

1. /3 Écrire une fonction `somme` d'argument `L` une liste et qui retourne la somme des éléments de la liste `L`.

```python
def somme(L:list) -> list:
    S = 0
    for x in L:
        S += x
    return S
```

2. Une matrice d'ordre n à coefficient entiers est un tableau à n lignes et n colonnes rempli d'entiers. Informatiquement, une matrice `M` d'ordre n peut être représentée par une liste de n listes de même taille, chaque liste correspondant à une ligne de la matrice.
Par exemple, si $M = \begin{pmatrix} 6 & 1 & 8 \\ 7 & 5 & 3 \\ 2 & 9 & 4 \end{pmatrix}$ est une matrice d'ordre 3, alors elle peut être représentée en Python par la liste de listes :

```python
M = [[6, 1, 8], [7, 5, 3], [2, 9, 4]]
```

Dans cet exemple, la première ligne est `M[0]` et `M[i][j]` permet d'accéder à l'élément situé à l'intersection de la ligne i et à la colonne j.

2.1) /3 En utilisant la fonction `somme`, écrire une fonction `som_ligne` d'argument `M` une matrice dimension $n \times n$ et qui renvoie la liste $[L_1, .., L_n]$ où pour tout i, $1 \leqslant i \leqslant n$, L_i représente la somme des éléments de la ligne i.

```python
def som_ligne(M:list)->list:
    L = []
    for ligne in M:
        L.append(somme(ligne))
    return L
```

```
>>> M = [[6, 1, 8], [7, 5, 3], [2, 9, 4]]
>>> som_ligne(M)
[15, 15, 15]
```

2.2) /4 Une matrice est dite *stochastique* si la somme sur chaque ligne est égale à 1. Par exemple, la matrice $M = \begin{pmatrix} 6 & 1 & 8 \\ 7 & 5 & 3 \\ 2 & 9 & 4 \end{pmatrix}$ n'est pas stochastique, alors que $M = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}$ l'est.
Compléter la fonction `est_stoch(M)` d'argument `M` une matrice pour qu'elle renvoie `True` si `M` est stochastique, et `False` dans le cas contraire. *(L'utilisation d'une boucle `while` est ici imposée afin d'éviter la présence d'un `return` dans une boucle `for`)*

```python
def est_sto(M:list)->bool:
    S = som_ligne(M)
    res = True
    k = 0
    n = len(M) # le nombre de lignes/colonnes
    while k < n and res:
        if S[k] != 1:
            res = False # on a trouvé une ligne de somme \
            ↪ != 1
        k = k+1
    return res
```

```
>>> M = [[6, 1, 8], [7, 5, 3], [2, 9, 4]]
>>> est_sto(M)
False
>>> M = [[1/3, 1/3, 1/3], [1, 0, 0], [1/2, 0, 1/2]]
>>> est_sto(M)
```

True

Interrogation d'ITC n°1B

Semaine du 09/10/2023

Nom :		Prénom :	

Consignes

- Les codes doivent être présentés en mentionnant explicitement l'indentation, au moyen par exemple de barres verticales sur la gauche.
- Pour qu'un code soit compréhensible, il convient de choisir des noms de variables les plus explicites possibles.

Exercice 3 **Étude de la 2-valuation d'un entier** [Sol 3] Pour tout entier $n \geq 1$, on désigne par $v(n)$ le plus grand entier k tel que 2^k divise n (c'est-à-dire tel que $\frac{n}{2^k} \in \mathbb{N}$). On a ainsi $v(2^3) = 3, v(12) = 2, v(3) = 0$.

1. On considère la fonction suivante :

```
1 def v(n:int)->int:
2     a = 1
3     k = 0
4     while n % a == 0 and a < n:
5         k = k+1
6         a = 2*a
7     return k-1
```

1.1) 2 Compléter le tableau ci-après, détaillant l'évolution des variables a et k lors de l'appel v(8).

a	k
1	0
2	1
4	2
8	3

Quel est le rôle de la variable a ? Quel est le rôle de la variable k ?

a correspond aux puissances de 2 que l'on teste comme diviseur de n, et k la puissance associée. En fin de boucle, on peut observer que k-1 vaut v(8)-1, on arrête donc plutôt la boucle lorsque a <= n

1.2) 2 Recopier et modifier le programme précédent afin qu'il retourne la valeur de $v(n)$, en effectuant uniquement une modification dans la ligne 4.

```
def v(n:int)->int:
    a = 1
    k = 0
    while n % a == 0 and a <= n:
        k = k+1
        a = 2*a
    return k-1
>>> v(8)
3
>>> v(7)
0
```

2. 3 Soit $N \in \mathbb{N}^\star$. On souhaite trouver la valuation minimale parmi les entiers de $[\![1, N]\!]$, c'est-à-dire $\min_{n \in [\![1,N]\!]} v(n)$. Compléter le code ci-après pour qu'il retourne cet entier.

```
def min_val(N : int) -> int:
    mini = v(1)
    for k in range(2, N+1) :
        if v(k) <= mini:
            mini = v(k)
    return mini
>>> min_val(10) # valeur que l'on peut contrôler en observant \
↪ la liste de la question suivante
0
```

3. 3 On peut montrer que l'on a pour $n \geq 1$:

$$v(n) = \begin{cases} 0 & \text{si } n \text{ est impair} \\ v(\frac{n}{2}) + 1 & \text{si } n \text{ est pair.} \end{cases}$$

Sans utiliser la fonction v, écrire une fonction `listeV(n)` d'argument n un entier supérieur ou égal à 1 et qui renvoie la liste contenant les valeurs $[v(1), ..., v(n)]$.

Il faut faire ici attention au décalage entre les indices de la liste $0, \dots, n-1$ et ceux de la suite v, qui sont $1, \dots, n$. Il faut aussi faire attention à utiliser // pour travailler avec des indices entiers.

```
def listeV(n:int)->list:
    L = []
    for i in range(1, n+1):
        if i % 2 == 1:
            # cas impair
            L.append(0)
        else:
            L.append(L[(i-1)//2]+1)
    return L
>>> listeV(10)
[0, 1, 0, 2, 0, 1, 0, 3, 0, 1]
>>> [v(i) for i in range(1, 11)]
[0, 1, 0, 2, 0, 1, 0, 3, 0, 1]
```

Exercice 4 **Matrices et listes de listes** [Sol 4] On rappelle un exemple d'utilisation de append dans la console.

```
>>> L = [1, 2, 3]
>>> L.append(1)
>>> L
[1, 2, 3, 1]
```

1. /3 Écrire une fonction somme d'argument L une liste et qui retourne la somme des éléments de la liste L.

```
def somme(L:list) -> list:
    S = 0
    for x in L:
        S += x
    return S
```

2. Une matrice d'ordre n à coefficient entiers est un tableau à n lignes et n colonnes rempli d'entiers. Informatiquement, une matrice M d'ordre n peut être représentée par une liste de n listes de même taille, chaque liste correspondant à une ligne de la matrice.
Par exemple, si $M = \begin{pmatrix} 6 & 1 & 8 \\ 7 & 5 & 3 \\ 2 & 9 & 4 \end{pmatrix}$ est une matrice d'ordre 3, alors elle peut être représentée en Python par la liste de listes :

```
M = [[6, 1, 8], [7, 5, 3], [2, 9, 4]]
```

Dans cet exemple, la première ligne est M[0] et M[i][j] permet d'accéder à l'élément situé à l'intersection de la ligne i et à la colonne j.

2.1) /3 En utilisant la fonction somme, écrire une fonction som_ligne d'argument M une matrice dimension $n \times n$ et qui renvoie la liste $[L_1, .., L_n]$ où pour tout i, $1 \leqslant i \leqslant n$, L_i représente la somme des éléments de la ligne i.

```
def som_ligne(M:list)->list:
    L = []
    for ligne in M:
        L.append(somme(ligne))
    return L
>>> M = [[6, 1, 8], [7, 5, 3], [2, 9, 4]]
>>> som_ligne(M)
[15, 15, 15]
```

2.2) /4 Une matrice est dite *stochastique* si la somme sur chaque ligne est égale à 1. Par exemple, la matrice $M = \begin{pmatrix} 6 & 1 & 8 \\ 7 & 5 & 3 \\ 2 & 9 & 4 \end{pmatrix}$ n'est pas stochastique, alors que $M = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}$ l'est.
Compléter la fonction est_stoch(M) d'argument M une matrice pour qu'elle renvoie **True** si M est stochastique, et **False** dans le cas contraire. *(L'utilisation d'une boucle **while** est ici imposée afin d'éviter la présence d'un **return** dans une boucle **for**)*

```
def est_sto(M:list)->bool:
    S = som_ligne(M)
    res = True
    k = 0
    n = len(M) # le nombre de lignes/colonnes
    while k < n and res:
        if S[k] != 1:
            res = False # on a trouvé une ligne de somme \
            ↪ != 1
        k = k+1
    return res
>>> M = [[6, 1, 8], [7, 5, 3], [2, 9, 4]]
>>> est_sto(M)
False
>>> M = [[1/3, 1/3, 1/3], [1, 0, 0], [1/2, 0, 1/2]]
>>> est_sto(M)
```

True