

Correction DS Informatique (concours blanc) durée 3h

Partie 1 Questions de cours

1. On traduit la relation donnée dans l'énoncé pour qu'elle donne u_n en fonction de u_{n-1} :

$$\forall n \geq 1, u_n = 2u_{n-1} - (n - 1)$$

En effet, dans l'écriture récursive de $u(n)$, on va devoir appeler récursivement $u(n-1)$. On a également le cas de base $u_0 = 2$, d'où le code :

```
def u(n) :  
    if n==0: # Cas de base  
        return 2  
    else: # Appel récursif  
        return 2*u(n-1) - (n-1)
```

► Explications 2, fonction correcte 2, cas de base 1, appel récursif 1 6

2. Dans le pire cas, pour trier un tableau de taille n :
- Le tri par insertion a une complexité temporelle $O(n^2)$;
 - Le tri fusion a une complexité temporelle $O(n \ln(n))$;
 - Le tri rapide (*quicksort*) a une complexité temporelle $O(n^2)$.

► 3 complexités et notation O 4

3. Le principe est de vider la pile P à l'aide d'une boucle *while* et d'utiliser une variable s pour retenir la somme des éléments retirés de P . En retirant les éléments de la pile P , on les place sur une autre pile Q ce qui permettra ensuite de reconstruire la pile P .

```
def somme(P) :  
    Q = pile_vide()  
    s = 0  
    while not est_vide(P) :  
        x = depiler(P)  
        empiler(Q, x)  
        s = s+x  
    while not est_vide(Q) :  
        empiler(P, depiler(Q))  
    return s
```

► Explications 2, fonction correcte 2, initialisation et return 1, première boucle 2, deuxième boucle 1 8

Partie 2 Bases de données

4.

```
SELECT NomFilm  
FROM films  
WHERE Budget >= 10000000
```

► Commande SQL 1

5.

```
SELECT NomRealisateur
FROM realisateurs
JOIN films ON films.IdRealisateur=realisateurs.IdRealisateur
WHERE Budget>=10000000
```

► **Commande SQL** [2] 1 seulement si erreur minime du type `IdRealisateur.films`

6.

```
SELECT SUM(Recettes),AnneeSortie GROUP BY AnneeSortie
FROM films
```

► **Commande SQL** [2] 1 seulement si erreur minime ou oubli de `AnneeSortie` dans **SELECT**

7.

```
SELECT NomFilm
FROM films
WHERE Recettes = (SELECT MAX(Recettes) FROM films)
```

► **Commande SQL** [3] 0 si **WHERE Recettes=MAX(Recettes)**

Partie 3 Modélisation

8. On pose $E(t) = v(t)^2 + \omega^2 x(t)^2$, on dérive :

$$\begin{aligned} E'(t) &= 2v'(t)v(t) + 2\omega^2 x'(t)x(t) = 2x''(t)x'(t) + 2\omega^2 x'(t)x(t) \quad \text{car } v(t) = x'(t) \text{ et } v'(t) = x''(t) \\ &= 2x'(t)(x''(t) + \omega^2 x(t)) = 0 \quad \text{car } x \text{ est solution de } (E) \end{aligned}$$

► **Calcul, conclusion** [2]

9. Par définition de $v(t)$, on a l'équation $x'(t) = v(t)$. De plus $x''(t) = v'(t)$ donc l'équation (E) s'écrit $v'(t) + \omega^2 x(t) = 0$. On en déduit le système d'ordre 1 :

$$\begin{cases} x'(t) = v(t) \\ v'(t) = -\omega^2 x(t) \end{cases}$$

► **Deux équations** [2]

10. On utilise la formule de Taylor-Young à l'ordre 2 :

$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + \underset{h \rightarrow 0}{o}(h) = x(t) + hv(t) - \frac{h^2}{2}\omega^2 x(t) + \underset{h \rightarrow 0}{o}(h)$$

Si Δ_t est assez petit, on aura donc $x(t + \Delta_t) \simeq x(t) + \Delta_t v(t) - \frac{\Delta_t^2}{2}\omega^2 x(t)$.

► **Développement limité, simplification, conclusion** [2]

11. Par exemple :

```
t = []
for i in range(0,N):
    t.append(i*Deltat)
```

► **Code correct** [2] 1 seulement si liste de taille $N + 1$

12. On traduit directement les relations, par exemple :

```
x = [x0]*N
v = [v0]*N
for i in range(0,N-1):
    x[i+1] = x[i]+Deltat*v[i]-Deltat**2/2*omega**2*x[i]
    v[i+1] = v[i]-Deltat/2*omega**2*(x[i]+x[i+1])
```

► **Code correct 2, initialisations 1, bornes du range 1, expressions 2** [6] -2 si c'est une fonction qui est écrite ou s'il y a un `return`

13. Dans la méthode d'Euler, on part du système d'ordre 1 :

$$\begin{cases} x'(t) = v(t) \\ v'(t) = -\omega^2 x(t) \end{cases}$$

et on utilise les approximations

$$x'(t) \simeq \frac{x(t + \Delta t) - x(t)}{\Delta t} \quad \text{et} \quad v'(t) \simeq \frac{v(t + \Delta t) - v(t)}{\Delta t}$$

ce qui donne $\frac{x_{i+1} - x_i}{\Delta t} = v_i$ et $\frac{v_{i+1} - v_i}{\Delta t} = -\omega^2 x_i$ ou encore :

$$x_{i+1} = x_i + \Delta t v_i \quad \text{et} \quad v_{i+1} = v_i - \Delta t \omega^2 x_i$$

► **Deux équations** [4] 1 s'il y a seulement les approximations des dérivées, -1 si les équations sont justes mais pas $x_{i+1} = \dots$, $v_{i+1} = \dots$

14. D'après la question 8, la quantité $E(t) = v(t)^2 + \omega^2 x(t)^2$ doit être constante au cours du temps. La représentation graphique montre que pour les solutions approchées obtenues numériquement, $E(t)$ varie beaucoup moins dans le cas de la méthode Leapfrog (entre $2 \pm 2 \cdot 10^{-5}$) que dans le cas de la méthode d'Euler (où $E(t)$ varie de 2 à 2.2). On en déduit que, sur ce critère, la solution approchée obtenue par la méthode Leapfrog constitue une meilleure approximation de la solution exacte que celle obtenue par la méthode d'Euler. *On peut aussi noter qu'avec la méthode Leapfrog, $E(t)$ varie (peu) mais oscille autour de la valeur 2, autrement $E(t)$ reste assez stable. Pour la méthode d'Euler, $E(t)$ augmente de manière (approximativement) linéaire.*

► **Explication raisonnable, conclusion** [3] 0 pour une conclusion non justifiée, ou justifiée de manière non raisonnable

Partie 4 Algorithmique

15. Ici $n = \text{len}(s) = 4$ et $m = \text{len}(t) = 3$. De plus, $s[0] = t[0]$, $s[1] = t[1]$ et $s[2] \neq t[2]$. En reprenant les notations de l'énoncé, on a $c = 1$ donc $\Delta(s, t) = 2$.

► **Réponse correcte** [1]

16. Si l'une des chaînes est vide, alors $c = 0$ donc $\Delta(s, t) = |n - m|$ en notant $n = \text{len}(s)$ et $m = \text{len}(t)$.

► **Réponse correcte** [1] 1 même si la réponse n'est que pour s (ou t) vide

17. On utilise un compteur c pour compter le nombre de différences $s[i] \neq t[i]$ pour $0 \leq i < \min(n, m)$ puis on ajoute $|n - m|$.

```
def delta(t, s):
    n = len(t)
    m = len(s)
    c = 0 # Nombre de différences
    i = 0
    while i < n and i < m:
        if t[i] != s[i]:
            c = c + 1
        i = i + 1
    return c + abs(n - m)
```

► Explications 2, code correct 2, initialisations 1, boucle (condition ou range) 1, corps de la boucle 1, valeur renvoyée 1 [8]

18. On peut passer de s à t en réalisant deux opérations : suppression du dernier caractère ce qui donne 'AB' puis remplacement du 'B' par un 'A'. Par ailleurs, on ne peut pas passer de s à t avec moins d'opérations car il faut au moins supprimer un caractère dans s et faire apparaître un 'A'. Ainsi, $\Delta(s, t) = 2$.

► Distance et justification avec deux opérations [2] Pas besoin de signaler qu'une seule opération ne suffit pas

19. Si s est vide et t est de longueur n , on peut obtenir t à partir de s en réalisant n insertions et on ne peut pas obtenir t avec moins d'opérations. On a donc $\Delta(s, t) = n$. De même, si s est de longueur n et t est vide, alors t est obtenue à partir de s en réalisant n suppressions et on ne peut pas l'obtenir avec moins d'opérations, donc $\Delta(s, t) = n$.

► Réponse correcte [1] 1 même si la réponse n'est que pour s (ou t) vide

20. On note toujours n la longueur de s et m la longueur de t . On utilise la propriété admise avec $i = n$ et $j = m$, la valeur renvoyée par la fonction est :

$$d[n, m] = \Delta_L(s[:n], t[:m]) = \Delta_L(s, t)$$

car on a $s[:n] = s$ et $t[:m] = t$.

► Utilisation de la propriété 1, $s[:n] = s$ et $t[:m] = t$ [2]

Remarque : une justification possible pour la propriété admise. On considère la propriété :

$$\mathcal{P}(i, j) : \ll d[i, j] = \Delta_L(s[:i], t[:j]) \gg$$

Appliquée avec $i = 0$, obtient avec la question 19 :

$$d[0, j] = \Delta_L(s[:0], t[:j]) = \Delta_L(' ', t[:j]) = j$$

et cette propriété est bien vraie une fois les deux premières boucles passées. De même, avec $j = 0$:

$$d[i, 0] = \Delta_L(s[:i], t[:0]) = \Delta_L(s[:i], '') = i$$

Cette propriété est bien vraie une fois les deux premières boucles passées. Par conséquent, la propriété $\mathcal{P}(i, j)$ est donc vraie lorsque $i = 0$ ou $j = 0$. Supposons maintenant que le propriété \mathcal{P} ne soit pas toujours vraie. On considère alors un couple (i_0, j_0) tel que $\mathcal{P}(i_0, j_0)$ est fausse. On choisit ce couple (i_0, j_0) de sorte que i_0 soit le plus petit possible (autrement dit i_0 est le premier numéro

de ligne pour lequel la propriété est fausse) et une fois i_0 fixé, on choisit également j_0 le plus petit possible. D'après ce qui précède, on a nécessairement $i_0 \geq 1$ et $j_0 \geq 1$ et on pose $i_0 = i + 1$ et $j_0 = j + 1$. D'après le choix de i_0 et j_0 , la propriété \mathcal{P} est vraie pour les couples $(i, j + 1)$, $(i + 1, j)$ et (i, j) . On note les chaînes :

$$\begin{aligned} s[i + 1] &= a_0 \cdots a_{i-1} a_i \\ t[j + 1] &= b_0 \cdots b_{j-1} b_j \end{aligned}$$

Si on note k le nombre minimal de transformations permettant de passer de $s[i + 1]$ à $t[j + 1]$, on distingue trois cas :

- Soit a_i est supprimé lors d'une transformation. Les $k - 1$ transformations qui restent consistent donc à transformer $a_0 \cdots a_{i-1}$ en $b_0 \cdots b_j$. Le nombre minimal d'opérations pour faire ceci est $d[i, j + 1]$ donc $k = d[i, j + 1] + 1$;
- Soit b_j est inséré lors d'une transformation. Les $k - 1$ transformations qui restent consistent donc à transformer $a_0 \cdots a_i$ en $b_0 \cdots b_{j-1}$. Le nombre minimal d'opérations pour faire ceci est $d[i + 1, j]$ donc $k = d[i + 1, j] + 1$;
- Dans les autres cas, le caractère a_i deviendra b_j ce qui ne demande aucune opération si $a_i = b_j$ et une opération si $a_i \neq b_j$. Les autres transformations effectuées reviennent à transformer $a_0 \cdots a_{i-1}$ en $b_0 \cdots b_{j-1}$. Le nombre minimal d'opérations pour faire ceci est $d[i, j]$ donc $k = d[i, j]$ si $a_i = b_j$ et $k = d[i, j] + 1$ sinon.

On pose $c = 1$ si $a_i \neq b_j$ et $c = 0$ sinon. On en déduit alors que :

$$d[i + 1, j + 1] = \min(d[i, j + 1] + 1, d[i + 1, j] + 1, d[i, j] + c)$$

est égal à $\Delta_L(a_0 \cdots a_i, b_0 \cdots b_j)$ d'où une contradiction. Donc il n'existe pas de couple (i, j) pour lequel \mathcal{P} n'est pas vérifiée.

21. (a) On a ici $n = \text{len}(s) = 3$ et $m = \text{len}(t) = 2$ donc d est un tableau à 4 lignes et 3 colonnes. Les deux premières boucles définissent $d[0, j] = j$ donc la première ligne de d est 0, 1, 2 et $d[i, 0] = i$ donc la première colonne de d est 0, 1, 2, 3. Ainsi, en ligne 11, d est la matrice :

$$d = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 0 \end{pmatrix}$$

► **Matrice correcte** 1

- (b) Il y a deux manières de répondre à cette question : soit on construit la matrice en suivant le déroulement du code, soit on utilise la propriété admise dans l'énoncé.

Méthode 1 : on suit le déroulement du code. Dans les lignes 12 à 18, la matrice d est remplie, ligne après ligne, en utilisant la formule de la ligne 18. Cette formule s'interprète de la manière suivante : la valeur de chaque case de d est définie comme le minimum des valeurs des 3 cases adjacentes (à gauche, en haut et en diagonale) en augmentant les valeurs des cases haut et gauche de 1 et en augmentant éventuellement la valeur de la case en diagonale de 1, lorsque les deux caractères dans s et t sont différents. On obtient après remplissage la matrice :

$$d = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 1 \\ 3 & 2 & 2 \end{pmatrix}$$

Méthode 2 : on utilise la propriété admise. D'après cette propriété, la matrice d est :

$$d = \begin{pmatrix} \Delta_L(s[:0], t[:0]) & \Delta_L(s[:0], t[:1]) & \Delta_L(s[:0], t[:2]) \\ \Delta_L(s[:1], t[:0]) & \Delta_L(s[:1], t[:1]) & \Delta_L(s[:1], t[:2]) \\ \Delta_L(s[:2], t[:0]) & \Delta_L(s[:2], t[:1]) & \Delta_L(s[:2], t[:2]) \\ \Delta_L(s[:3], t[:0]) & \Delta_L(s[:3], t[:1]) & \Delta_L(s[:3], t[:2]) \end{pmatrix}$$

Pour que ce soit plus clair, on écrit explicitement les chaînes :

$$d = \begin{pmatrix} \Delta_L(' ', ' ') & \Delta_L(' ', 'A') & \Delta_L(' ', 'AA') \\ \Delta_L('A', ' ') & \Delta_L('A', 'A') & \Delta_L('A', 'AA') \\ \Delta_L('AB', ' ') & \Delta_L('AB', 'A') & \Delta_L('AB', 'AA') \\ \Delta_L('ABC', ' ') & \Delta_L('ABC', 'A') & \Delta_L('ABC', 'AA') \end{pmatrix}$$

On peut déjà simplifier dans le cas où l'une des chaînes est vide :

$$d = \begin{pmatrix} 0 & 1 & 2 \\ 1 & \Delta_L('A', 'A') & \Delta_L('A', 'AA') \\ 2 & \Delta_L('AB', 'A') & \Delta_L('AB', 'AA') \\ 3 & \Delta_L('ABC', 'A') & \Delta_L('ABC', 'AA') \end{pmatrix}$$

Et pour les éléments restants, il est assez simple de calculer la distance. On obtient :

$$d = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 1 \\ 3 & 2 & 2 \end{pmatrix}$$

► **Matrice correcte** [2] 1 si le principe est correct

22. La première boucle a un temps d'exécution en $O(n)$, la deuxième boucle a un temps d'exécution en $O(m)$. Ensuite on a une boucle de taille m imbriquée dans une boucle de taille n , le temps d'exécution est donc $O(nm)$. Finalement, le temps d'exécution de **deltaL** est $O(n) + O(m) + O(nm) = O(nm)$.

► **Complexité temporelle 1, justification (même très rapide)** 1 [2]

23. La fonction **deltaL** utilise essentiellement un tableau de taille $n \times m$, sa complexité en espace est donc $O(nm)$.

► **Complexité en espace 1, justification (même très rapide)** 1 [2]

24. On remarque que dans la construction de la matrice **d**, on a seulement besoin de la ligne courante et de la ligne précédente. On utilise donc (à la place de **d**) deux variables **dcour** qui représente la ligne $i + 1$ de **d** et **dprec** qui représente la ligne i de **d**. On reprend le code de **deltaL** en remplaçant alors **d[i+1, j]** par **dcour[j]** et **d[i, j]** par **dprec[j]**. Il faut aussi initialiser correctement **dcour**.

```

def deltaL2(s,t):
    n = len(s)
    m = len(t)
    dcour = np.zeros(m+1)
    for j in range(0,m):
        dcour[j+1] = j+1
    for i in range(0,n):
        dprec = dcour
        dcour = np.zeros(m+1)
        dcour[0] = i+1
        for j in range(0,m):
            if s[i]==t[j]:
                rempl = 0
            else:
                rempl = 1
            dcour[j+1] = min(dprec[j+1]+1,dcour[j]+1,dprec[j]+rempl)
    return dcour[m]

```

► Explications 2, code correct 2, utilisation de 2 lignes 1, reste du code 3 10

Sur l'ensemble de la copie :

► Respect de la syntaxe Python 1

► Présentation 1