Correction

NSI - 2021 Sujet Zéro (21-sujet-zero)

Exercice 1 - Pile de crêpes

1. Le contenu de la pile Q après l'exécution des instructions est :

8
5
2
4

2.1.

```
def hauteur_pile(P):
    Q = creer_pile_vide()
    n = 0
    while not(est_vide(P)):  # Compter
        n += 1
        x = depiler (P)
        empiler(Q, x)
    while not(est_vide(Q)):  # Rétablir la pile
        x = depiler(Q)
        empiler(P, x)
    return n
```

2.2.

```
def max_pile(P, i):
    """ Recherche de la position du maximum parmi les i derniers éléments
      de la pile P. Le sommet de la pile est 1. P conserve son état d'origine. """
    if est_vide(P):
        return "??"
    j, j_max = 1, 1
    Q = creer_pile_vide()
    val_max = depiler(P)
    i -= 1
    empiler(Q, val_max)
    while not est_vide(P) and i > 0 :
        j += 1
        i -= 1
        val = depiler(P)
        empiler(Q, val)
        if val > val_max:
            val_max = val
            j_max = j

    while not est_vide(Q): # Rétablir la pile P
        empiler(P, depiler (Q))

    return j_max
```

Correction

NSI - 2021 Sujet Zéro (21-sujet-zero)

3.

```
def retourner(P, j):
    Q = creer_pile_vide()
    R = creer_pile_vide()
    while j > 0:  # Retirer les j premiers éléments
        empiler(Q, depiler(P))
        j -= 1
    while not est_vide(Q):  # Déplacer dans la pile R
        empiler(R, depiler(Q))
    while not est_vide(R):  # Remettre dans la pile P
        empiler(P, depiler(R))
```

4.

```
def tri_crepes(P):
    for i in range(hauteur_pile(P), 1, -1):
        j = max_pile(P, i) # Recherche le plus grand
        retourner(P, j)    # Retourne jusqu'au plus grand
        retourner(P, i)    # Retourne la pile entière restant à trier
```

Correction

NSI - 2021 Sujet Zéro (21-sujet-zero)

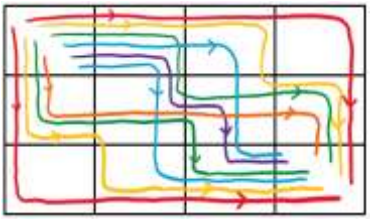
Exercice 2 - Chemin de somme maximale

1.1. Pour atteindre la case (2, 3) en partant de (0, 0), il y a nécessairement 2 déplacements vers le bas.

1.2. Chaque chemin a 2 cases vers le bas, 3 vers la droite, sans oublier la case de départ, cela mène à une longueur de 6 cases.

2. Les chemins possibles sont :

Chemin	Somme
(0, 0) → (0, 1) → (0, 2) → (0, 3) → (1, 3) → (2, 3)	11
(0, 0) → (0, 1) → (0, 2) → (1, 2) → (1, 3) → (2, 3)	10
(0, 0) → (0, 1) → (1, 1) → (1, 2) → (1, 3) → (2, 3)	9
(0, 0) → (0, 1) → (0, 2) → (1, 2) → (2, 2) → (2, 3)	14
(0, 0) → (0, 1) → (1, 1) → (1, 2) → (2, 2) → (2, 3)	13
(0, 0) → (1, 0) → (1, 1) → (1, 2) → (1, 3) → (2, 3)	10
(0, 0) → (0, 1) → (1, 1) → (2, 1) → (2, 2) → (2, 3)	12
(0, 0) → (1, 0) → (1, 1) → (1, 2) → (2, 2) → (2, 3)	14
(0, 0) → (1, 0) → (1, 1) → (2, 1) → (2, 2) → (2, 3)	13
(0, 0) → (1, 0) → (2, 0) → (2, 1) → (2, 2) → (2, 3)	16



4	1	1	3
2	0	2	1
3	1	5	1

Le chemin qui permet d'obtenir la somme maximale (16) est le chemin : (0, 0) → (1, 0) → (2, 0) → (2, 1) → (2, 2) → (2, 3).

3. 1. Tableau des sommes maximales complétés :

4	5	6	**9**
6	**10**	8	10
9	10	**15**	16

3.2. $T'[0][j] = T[0][j] + T'[0][j-1]$ est vrai pour j différent de 0 car cela revient à parcourir la première ligne du tableau. En effet, pour obtenir la valeur d'une case $(0, j)$, il faut faire l'addition de la valeur de cette case avec la somme maximale déjà calculée dans la case précédente $(0, j-1)$.

4. Nous recherchons le chemin réalisant la somme maximale pour aller sur une case en venant du haut ou de la gauche. Ainsi, il faut ajouter la valeur de cette case $(i\ , j)$ à la plus grande somme déjà calculée sur la case du dessus $(i-1, j)$ ou sur celle de gauche $(i, j-1)$.

5.1. Le cas de base (ou cas trivial) est celui de la case $(0, 0)$ ou l'on retourne directement la valeur 4.

5.2.

```
def somme_max(T, i, j):
    if i == 0 and j == 0:  # Cas de base
        return T[0][0]
    elif i == 0:           # Cas de la première ligne (récursif simple)
        return T[i][j] + somme_max(T, i, j-1)
    elif j == 0 :          # Cas de la première colonne (récursif simple)
        return T[i][j] + somme_max(T, i-1, j)
    else :                 # Cas général (récursif double)
        return T[i][j] + max(somme_max(T, i-1, j), somme_max(T, i, j-1))
```

5.3. Pour résoudre le problème il faut faire l'appel : `somme_max(T, len(T)-1, len(T[0])-1)`

Correction

NSI - 2021 Sujet Zéro (21-sujet-zero)

Exercice 3 - Arbre binaire de recherche

Attention : La convention choisie dans l'exercice dit que la hauteur d'un arbre binaire ne comportant qu'un nœud (c'est-à-dire seulement sa racine) est 1. Ici la hauteur correspond au nombre de niveau de l'arbre.

1. La taille correspond au nombre de nœud. Cet arbre est de taille 9.
Sa hauteur (nombre de niveaux) est de 4.

2. 1. Le numéro binaire associé au nœud G est 101.

2. 2. $13_{(10)} = 1101_{(2)}$ ce qui correspond au nœud I

2. 3. Les nœuds de hauteur 2 sont numérotés sur 2 bits, ceux de hauteur 3 sur 3 bits, … ceux de hauteur h, sur h bits.

2. 4. Au minimum, pour un arbre complètement dégénéré (linéaire), il n'y a qu'un nœud par niveau soit $n = h$. Tandis que pour un arbre binaire complet, il y a $n = 2^h - 1$ nœuds car sur h bits, il y a 2^h valeurs possibles. Le -1 provient du fait qu'il n'y a pas de nœud portant la valeur 0.
Ainsi, n est borné par :

$$h \leq n \leq 2^h - 1$$

3. 1. Tableau représentant l'arbre binaire complet :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

3. 2. Le père du nœud d'indice i avec $i \geq 2$ a pour indice la partie entière de $i/2$.

4. Les arbres binaires de recherche permette d'implémenter l'algorithme de recherche par dichotomie.

```
def recherche(arbre, element):
    """ Recherche par dichotomie """
    i = 1
    while i < len(arbre):  # Tant que l'on est dans le tableau
        if element == arbre[i]:
            return True
        elif element < arbre[i]:  # Recherche dans le sous-arbre gauche
            i = 2*i
        else:  # Recherche dans le sous-arbre droit
            i = 2*i + 1
    return False
```

Correction

NSI - 2021 Sujet Zéro (21-sujet-zero)

Exercice 4 - Base de données eleve

1. 1. L'attribut num_eleve est une clé primaire. Il permet d'identifier de manière unique chaque enregistrement de la relation (table) et permet aussi d'assurer la contrainte de relation qui est une des trois contraintes d'intégrité des bases de donnes (avec les contraintes de référence et de domaine).

1. 2. Requête d'insertion :

```
INSERT INTO seconde(num_eleve, langue1, langue2, classe)
VALUES(133310, 'anglais', 'espagnol', '2A') ;
```

ou

```
INSERT INTO seconde VALUES(133310, 'anglais', 'espagnol', '', '2A') ;
```

Remarque : L'énoncé précise que l'attribut num_eleve est un entier bien qu'il y ait deux lettres dans les valeurs de la figure 1.

1. 3. Requête de mise à jour :

```
UPDATE seconde SET langue1 = 'allemand' WHERE num_eleve = 156929
```

2. 1. Cette requête affiche toutes les valeurs de l'attributs num_eleve.

2. 2. Cette requête donne le nombre d'enregistrements (d'élève) dans la table.

2. 3. Requête d'interrogation :

```
SELECT COUNT(num_eleve) FROM seconde
WHERE langue1 = 'allemand' OR langue2 = 'allemand' ;
```

3. 1. Une clé étrangère fait référence à une clé primaire d'une autre table. La cohérence d'une base de données est assurée par les contraintes d'intégrité. Une clé étrangère permet d'implémenter la contrainte de référence qui empêche de créer un enregistrement faisant référence à un enregistrement qui n'existe pas.

3. 2.

```
SELECT eleve.nom, eleve.prenom, eleve.datenaissance
FROM eleve JOIN seconde ON eleve.num_eleve = seconde.num_eleve
WHERE seconde.classe = '2A'
```

ou

```
SELECT eleve.nom, eleve.prenom, eleve.datenaissance
FROM eleve, seconde
WHERE eleve.num_eleve = seconde.num_eleve AND seconde.classe = '2A'
```

4.

coordonnees
num_eleve (clé primaire, clé étrangère de la table seconde)
adresse
code_postal
ville
adresse_mail

Correction

NSI - 2021 Sujet Zéro (21-sujet-zero)

Exercice 5 - Protocole de routage

1. 1. D'après les tables de routage, le chemin obtenu par le protocole RIP est :
A → C → F → G avec 3 sauts (3 hops)

1. 2.

Table de routage du routeur G		
Destination	Routeur Suivant	Distance (Métrique)
A	E	3
B	E	3
C	E	2
D	E	2
E	E	1
F	F	1

2. Panne du routeur C

Table de routage du routeur A		
Destination	Routeur Suivant	Distance (Métrique)
B	B	1
C	**?**	**?**
D	D	1
E	**D**	2
F	**D**	**4**
G	**D**	**3**

3. 1. Calcul du coût de la liaison entre A et B :

$$coût = \frac{10^8}{d} = \frac{10^8}{10.10^9} = 0{,}01$$

3. 2. Calcul du débit de la liaison entre B et D :

$$d = \frac{10^8}{coût} = \frac{10^8}{5} = 20.10^6 = 20\ Mbits/s$$

4. Le chemin parcouru selon le protocole OSPF (minimisation du coût) est :
A → D → E → G (coût de 1,011)