Amérique du nord - mai 2024 - sujet 2

Exercice 1 (Programmation Python et algorithmes de tri - 6 points)
On se propose dans cet exercice de se pencher sur un algorithme pour trier un tableau appelé le *tri de Stooge*.

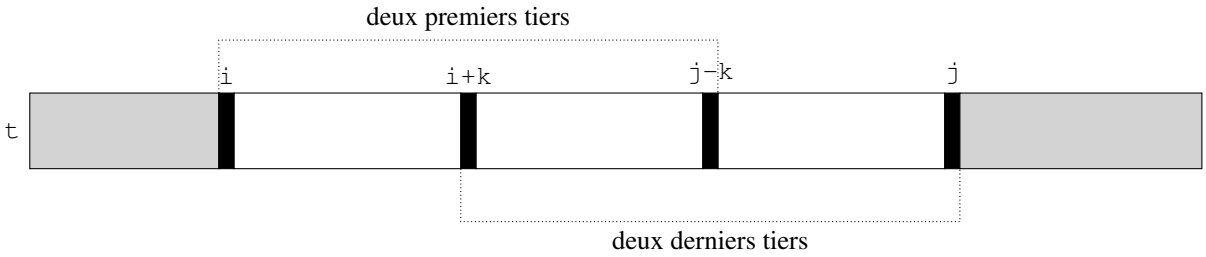
Pour trier les éléments situés entre les indices `i` et `j`, où `i < j`, dans un tableau `t` par ce tri, on procède ainsi :

- si les éléments d'indice `i` et `j` sont mal placés, on les échange ;
- s'il y a au moins trois éléments entre les indices `i` et `j` :
  - on trie les deux premiers tiers du tableau avec cette méthode ;
  - on trie les deux derniers tiers du tableau avec cette méthode ;
  - on trie à nouveau les deux premiers tiers du tableau avec cette méthode.

Pour réaliser ce découpage en tiers, on considère l'entier `k` défini par l'expression

```
k = (j - i + 1)//3
```

et on considère les indices intermédiaires `i+k` et `j-k`.



Voici le code partiel de l'algorithme du tri de Stooge en Python qui trie donc les éléments d'un tableau par ordre croissant :

```
1  def triStooge(tab, i, j):
2      if tab[i] > tab[j]:
3          echange(tab, i, j)
4      if (j - i) > 1:
5          k = (j - i + 1)//3
6          triStooge(...)
7          triStooge(...)
8          triStooge(...)
```

1. Ecrire la fonction `echange(tab, i, j)` qui prend en arguments une liste Python `tab` et deux indices `i` et `j`, et qui réalise sur place l'échange des valeurs dans `tab` à ces indices. La fonction ne renvoie rien.
2. Finaliser le programme précédent en complétant les lignes 6, 7 et 8 sur votre copie.
3. Indiquer en le justifiant si cet algorithme est itératif ou récursif.

Soit l'appel `triStooge(A, 0, 5)` avec `A = [5, 6, 4, 2, 3, 1]`.

4. Déterminer la valeur numérique prise par `k` lors de ce premier appel. Une justification est attendue.

La figure ci-dessous présente l'arbre (incomplet) des appels récursifs effectués depuis l'appel `triStooge(A, 0, 5)`.

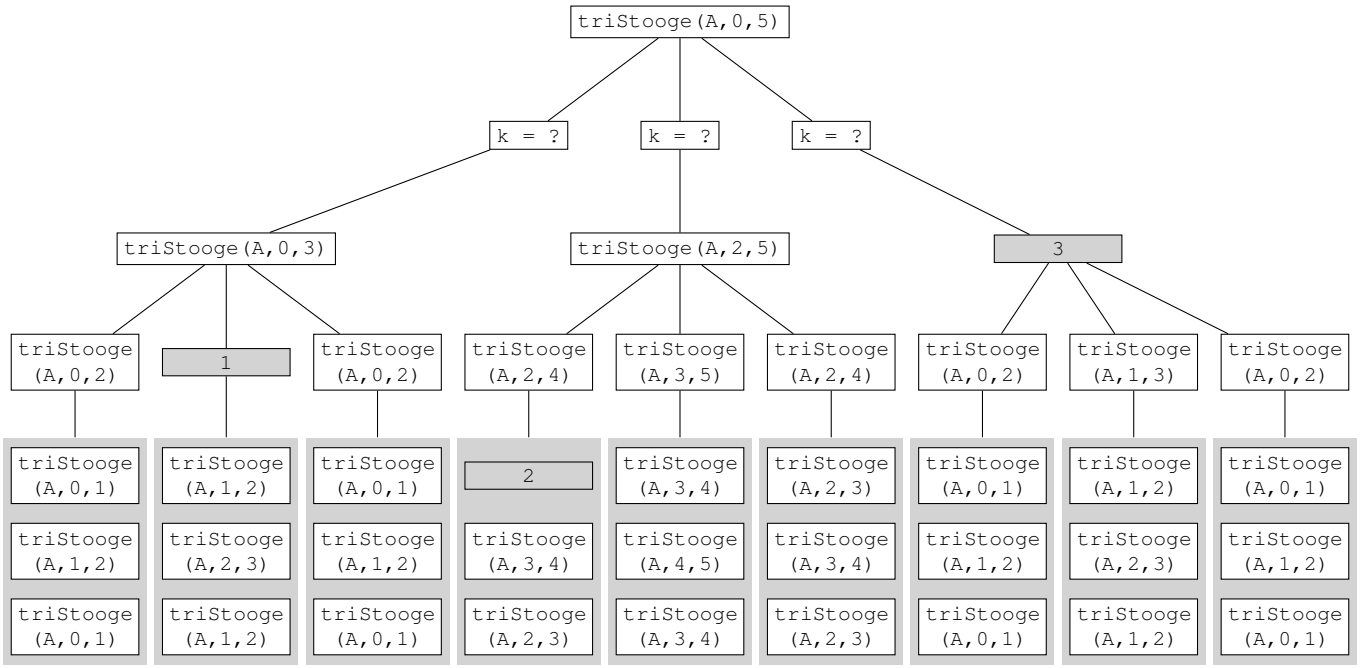


FIGURE 1 – Arbre des appels récursifs pour `triStooge(A, 0, 5)`

5. Dénombrer le nombre d'appels récursifs effectués lors du tri sans compter l'appel initial.
6. Déterminer les appels effectués dans les cases 1, 2 et 3 de cet arbre des appels récursifs.

On montre que le coût en temps dans le pire des cas de l'algorithme de Stooge est de l'ordre de n^e avec e environ égal à $\frac{8}{3}$.

7. Donner un algorithme de tri donc le coût est strictement meilleur.

Exercice 2 (Bases de données et SQL - 6 points)
Un pharmacien nouvellement installé décide de créer son propre système de gestion des médicaments qu'il délivre à ses clients.

Pour sa base de données relationnelle, il a déjà élaboré la première relation à l'aide des données indiquées sur les cartes vitales de ses deux premiers clients :
`client(id_client: INT, nom_client: VARCHAR(30), prenom_client: VARCHAR(30), num_secu_sociale: VARCHAR(15))`

client			
id_client	nom_client	prenom_client	num_secu_sociale
1	Martin	Sophie	202103812326129
2	Dufour	Marc	105073817009595

1. Ecrire le résultat de l'exécution de la requête SQL suivante :

```
SELECT nom_client, prenom_client FROM client ORDER BY nom_client
```

Pour écrire la relation `medicament`, il doit utiliser les informations fournies par la notice des médicaments. En voici une ci-dessous :

Paracétamol 1 gramme CP

Qu'est-ce que **Paracétamol 1 gramme CP** et dans quel cas est-il utilisé ?
Paracétamol 1 gramme CP est un antalgique (calme la douleur)
Que contient un comprimé de **Paracétamol 1 gramme CP** ?
La substance active est le paracétamol : 1 gramme pour un comprimé
Sous quelle forme se présente **Paracétamol 1 gramme CP** ? Ce médicament se présente sous la forme de comprimé.
Chaque boîte contient 8 comprimés.

FIGURE 1 – Informations extraites de la notice du médicament Paracétamol 1 gramme CP

La relation `medicament` suivante a été obtenue à l'aide de ces notices :
`medicament(id_medic: INT, nom_medic: VARCHAR(30), categorie: VARCHAR(20), conditionnement: INT, quantite: INT, prix: FLOAT)` La table des médicaments de son officine est présentée ci-dessous :

medicament					
id_medic	nom_medic	categorie	conditionnement	quantite	prix
1	Paracétamol 1 gramme CP	antalgique	8	50	3,50
2	Acide acétylsalicylique	antalgique	8	20	2,30
3	Gel hydroalcoolique 100 ml	désinfectant	1	300	2,30
4	Acide ascorbique	vitamine	10	450	5,50

2. Ecrire une requête SQL permettant d'afficher les noms de tous les médicaments dont le prix est strictement inférieur à 3 euros.

Madame Martin présente au pharmacien une nouvelle ordonnance :

```
                                CENTRE MEDICAL

Dr Louis FARTI
           Grenoble, le 13 décembre 2023



                                Mme Sophie MARTIN


  — Paracétamol 1 gramme CP (boîte de 8)
    Prendre, par voie orale, 1 comprimé par prise, à renouveler en cas de besoin au
    bout de 4 heures minimum, avec au maximum 3 comprimés par jour, pendant 2 jours.

  — Acide ascorbique (vitamine C) 500 mg comprimé effervescent (boîte de 10)
    Un comprimé par jour pendant 4 semaines.
```

FIGURE 2 – Ordonnance de Madame Sophie Martiin

Il saisit les informations de cette ordonnance dans la relation `ordonnance`, chaque médicament prescrit correspondant à un enregistrement dans la table ci-dessous :

ordonnance				
id_ordo	id_client	date_ordo	id_medic	nb_boites
6	2	2023-11-29	2	2
7	1	2023-12-13	1	...
8	1	2023-12-13	4	...

3. Ecrire une requête SQL permettant d'ajouter les informations de la carte vitale de sa troisième cliente présentée ci-dessous :

FIGURE 3 – Image de la carte vitale extraite de la page wikipédia https://fr.wikipedia.org/wiki/Carte_Vitale

4. Donner les attributs qui doivent être déclarés comme clés étrangères de la relation `ordonnance` et en préciser l'utilisé.
5. Indiquer, pour les lignes 7 et 8 de la table `ordonnance`, le nombre de boîtes prescrites.
6. Ecrire la requête SQL mettant à jour la quantité du médicament Acide ascorbique en stock dans l'officine du pharmacien suite au passage de Madame Martin.
7. Calculer le coût total des médicaments fournis à Madame Martin (on ne demande pas d'écrire une requête ici, mais de calculer le coût total en justifiant le calcul).
8. Ecrire la requête SQL permettant d'afficher le nom du médicament pour l'ordonnance ayant l'`id_ordo` numéro 6.

Exercice 3 (POO, listes chaînées, réseaux, architecture matérielle - 8 points)

On considère un réseau local constitué des trois machines de Alice, Bob et Charlie dont les adresses IP sont les suivantes :

- la machine d'Alice a pour adresse `192.168.1.1` ;
- la machine de Bob a pour adresse `192.168.1.2`.

On rappelle que l'adresse `192.168.1.255` est l'adresse de diffusion qui sert à communiquer avec toutes les machines du réseau local et que le masque de ce réseau local est `255.255.255.0`. Cette adresse de diffusion est réservée et ne peut être attribuée à une machine.

Partie A.

1. Donner une adresse IP possible pour la machine de Charlie afin qu'elle puisse communiquer avec celles d'Alice et Bob dans le réseau local. Justifier votre réponse en donnant toutes les conditions à respecter dans le choix de cette adresse IP.

Ce réseau est utilisé pour effectuer des transactions financières en monnaie numérique `nsicoin` entre les trois utilisateurs. Pour cela, on crée la classe `Transaction` ci-dessous :

```
class Transaction:
    def __init__(self, expediteur, destinataire, montant):
        self.expediteur = expediteur
        self.destinataire = destinataire
        self.montant = montant
```

2. Toutes les dix minutes, les transactions réalisées pendant cet intervalle de temps sont regroupées par ordre d'apparition dans une liste Python. Dans un intervalle de dix minutes, Alice envoie dix `nsicoin` à Charlie, puis Bob envoie cinq `nsicoin` à Alice. Ecrire la liste Python correspondante à ces transactions.

Pour garder une trace de toutes les transactions effectuées, on utilise une liste chaînée de blocs (ou *blockchain*) dont le code Python est fourni ci-dessous. Toutes les dix minutes, un nouveau bloc contenant les nouvelles transactions est créé et ajouté à la *blockchain*.

```
class Bloc:
    def __init__(self, liste_transactions, bloc_precedent):
        self.liste_transactions = liste_transactions
        self.bloc_precedent = bloc_precedent # de type Bloc

class Blockchain:
    def __init__(self):
        self.tete = self.creer_bloc_0()

    def creer_bloc_0(self):
        """
        Crée le premier bloc qui distribue 100 nsicoin à tous les utilisateurs
        (un pseudo-utilisateur Genesis est utilisé comme expéditeur)
        """
        liste_transactions = [
            Transaction("Genesis", "Alice", 100),
            Transaction("Genesis", "Bob", 100),
            Transaction("Genesis", "Charlie", 100)
        ]
        return Bloc(liste_transactions, None)
```

3. La figure 1 représente les trois premiers blocs d'une Blockchain. Expliquer pourquoi la valeur de l'attribut `bloc_precedent` du `bloc0` est `None`.

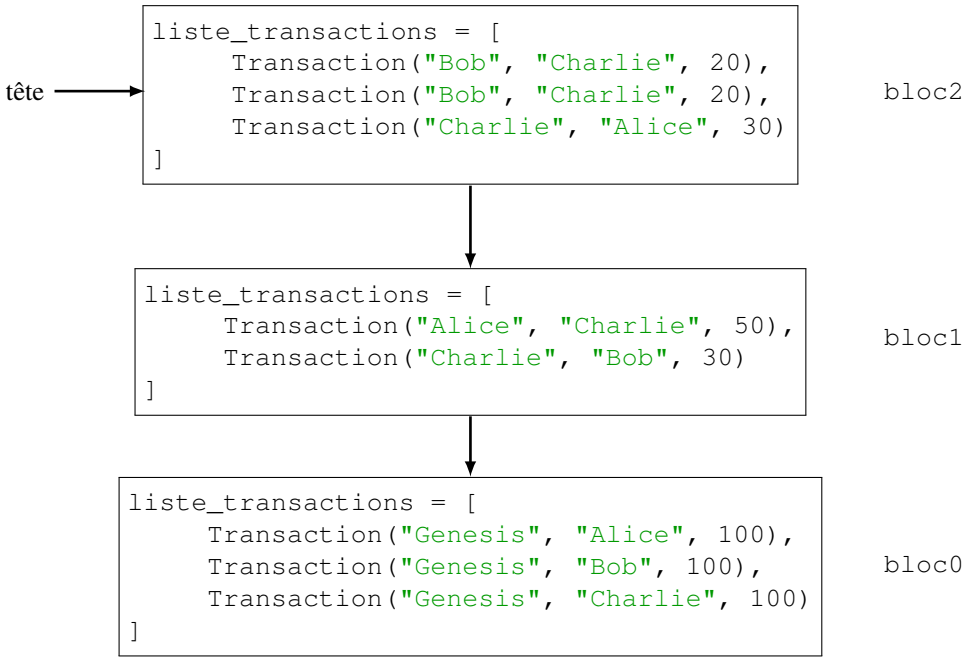


FIGURE 1 – Blockchain

4. Donner la valeur de l'attribut `bloc_precedent` du `bloc1` afin que celui-ci soit lié au `bloc0`.
5. A l'aide des classes `Bloc` et `Blockchain`, écrire le code Python permettant de créer un objet `ma_blockchain` de type `Blockchain` représenté par la figure 1.
6. Donner le solde en `nsicoin` de Bob à l'issue du `bloc2`.
7. On souhaite doter la classe `Blockchain` d'une méthode `ajouter_bloc` qui prend en paramètre la liste des transactions des dix dernières minutes et l'ajoute dans un nouveau bloc. Ecrire le code Python de cette méthode ci-dessous :

```
1 def ajouter_bloc(self, liste_transactions):
2     # A compléter
```

8. Lorsqu'un utilisateur ajoute un nouveau bloc à la Blockchain, il l'envoie aux autres membres. Ainsi, chaque utilisateur dispose sur sa propre machine d'une copie identique de la blockchain. Donner le nom et la valeur de l'adresse IP à utiliser pour effectuer cet envoi.
9. On souhaite doter la classe `Bloc` d'une nouvelle méthode `calculer_solde` permettant de renvoyer le solde à l'issue de ce bloc. Recopier et compléter sur votre copie le code Python de cette méthode :

```
1  def calculer_solde(self, utilisateur):
2      if self.bloc_precedent is None: # cas de base
3          solde = 0
4      else:
5          solde = ... # appel récursif : calcul du solde au bloc précédent
6          for transaction in self.liste_transactions:
7              if ... == utilisateur:
8                  solde = solde - ...
9              elif ... :
10                 ...
11         return solde
```

10. Ecrire l'appel à la fonction `calculer_solde` permettant de calculer le solde actuel d'Alice.

Partie B.

Dans cette partie, on va améliorer la sécurité de la blockchain. Pour cela, on enrichit la classe `Bloc` comme indiquée ci-dessous :

```
class Bloc:
    def __init__(self, liste_transactions, bloc_precedent):
        self.liste_transactions = liste_transactions
        self.bloc_precedent = bloc_precedent
        # Définition de trois nouveaux attributs
        self.hash_bloc_precedent = self.donner_hash_precedent()
        self.nonce = 0 # fixé arbitrairement et temporairement à 0 avant le minage du bloc
        self.hash = self.calculer_hash()

    # Définition de trois nouvelles méthodes
    def donner_hash_precedent(self):
        if self.bloc_precedent is not None:
            return self.bloc_precedent.hash
        else:
            return "0"

    def calculer_hash(self):
        """calcule et renvoie le hash du bloc courant"""
        # Le code Python n'est pas étudié dans cet exercice

    def minage_bloc(self):
        """modifie le nonce d'un bloc pour que son hash commence par '00'"""
        # A compléter
```

La fonction `calculer_hash` produit une chaîne de caractères appelée `hash` qui possède les propriétés suivantes :

- le hash d'un bloc dépend de toutes les données contenues dans le bloc et uniquement de ces données ;
- le calcul du hash d'un bloc est rapide et facile à calculer par une machine ;
- la moindre modification dans le bloc produit un hash complètement différent ;
- il est impossible de déduire le bloc à partir de son hash ;
- si deux blocs ont le même hash, c'est qu'ils sont parfaitement identiques.

11. L'attribut `nonce` est de type entier. Miner un bloc signifie trouver une valeur de `nonce` de telle façon que l'attribut `hash` du bloc commence par les deux caractères `'00'`. Compte-tenu des propriétés précédentes, la seule façon de trouver cette valeur est de procéder à une recherche exhaustive. Expliquer en quoi consiste le fait de trouver une valeur par recherche exhaustive.

Dans la suite de l'exercice, on considère que tous les utilisateurs cherchent à miner un nouveau bloc. Le premier qui réussit l'ajoute à la blockchain et gagne une récompense en `nsicoin`.

12. En justifiant votre réponse, donner la valeur de l'attribut `hash_bloc_precedent` du `bloc0`.
13. Sachant que le `hash` est écrit sur 256 bits, donner le calcul permettant d'obtenir le nombre de hash possibles.
14. Recopier et compléter sur votre copie le code Python ci-dessous de la méthode `minage_bloc`.

```
def minage_bloc(self):
    """modifie le nonce d'un bloc pour que son hash commence par '00' en
    énumérant tous les entiers naturels en partant de 0."""
    self.nonce = 0
    self.hash = self.calculer_hash()
    while ... :
        self.nonce = ...
        self.hash = ...
```