Baccalauréat général

Epreuve blanche d’enseignement de spécialité

JANVIER 2025

NUMÉRIQUES ET SCIENCES INFORMATIQUES

Durée de l’épreuve 3 heures 30

Aucun document autorisé

L’usage de la calculatrice n’est pas autorisé.

Dès que ce sujet est remis, assurez-vous qu’il est complet

Le sujet n’est pas à rendre avec la copie

**Le sujet est composé de trois exercices indépendants**

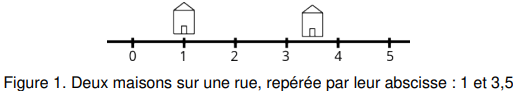
# Exercice 1 (6 points)

Cet exercice porte sur la programmation Python, la programmation orientée objet et l’algorithmique.

Une entreprise doit placer des antennes relais le long d’une rue rectiligne. Une antenne relais de portée (ou rayon) couvre toutes les maisons qui sont à une distance inférieure ou égale à de l’antenne.

Connaissant les positions des maisons dans la rue, l’objectif est de placer les antennes le long de la rue, pour que toutes les maisons soient couvertes, tout en en minimisant le nombre d’antennes utilisées.

La rue est représentée par un axe, et les maisons sont représentées des points sur cet axe :



Les entités manipulées sont modélisées en utilisant la programmation orientée objet.

class Maison:

    def \_\_init\_\_(self, position):

        self.\_position = position

    def get\_pos\_maison(self):

        return self.\_position

class Antenne:

    def \_\_init\_\_(self, position, rayon):

        self.\_position = position

        self.rayon = rayon

    def get\_pos\_antenne(self):

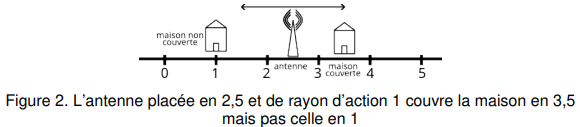
        return self.\_position

    def get\_rayon(self):

        return self.rayon

1. Donner le code qui crée et initialise deux variables et avec des instances de la classe situées aux abscisses 1 et 3,5 (Figure 1)

On ajoute à présent une antenne ayant un rayon d’action de 1 à la position 2,5 :



1. Donner le code qui crée la variable correspondant à l’antenne à la position avec le rayon d’action .

On souhaite modéliser une rue par une liste d’objets de type . Cette liste sera construite à partir d’une autre liste contenant des nombres correspondant aux positions des maisons. La fonction réalise ce travail. Elle prend en paramètre une liste de positions et renvoie une liste d’objets de type .

1. Recopier le schéma ci-dessous et le compléter pour donner une représentation graphique de la situation créée par :

creation\_rue([0, 2, 3, 4, 5, 7, 9, 10.5, 11.5])

1. Compléter le code donné ci-dessous de la fonction .

def creation\_rue(pos):

    pos.sort()

    maisons = []

    for p in pos:

        m = Maison(p)

        maisons.append(...)

    return ...

Pour rappel : la commande trie la liste .

La méthode de la classe prend en paramètre un objet de type et indique par un booléen si l’antenne couvre la maison en question ou non. La méthode peut être utilisée ainsi (en supposant que les objets précédents m1, m2 et a existent).

>>> a.couvre(m1)

False

>>> a.couvre(m2)

True

1. Compléter la fonction , ci-dessous, en veillant à ne pas accéder directement aux attributs d’une maison depuis la classe Antenne (on pourra utiliser les méthodes et ).

Pour rappel : la fonction valeur absolue se nomme en Python.

    # Méthode à ajouter dans la classe Antenne

    def couvre(self, maison):

        # Code à compléter (éventuellement plusieurs lignes)

        ...

La fonction est donnée ci-dessous. L’objectif est de placer des antennes dans une rue. Elle est fournie à la société qui place les antennes. La fonction prend en paramètre une liste d’objets de type (qu’on supposera triée par abscisse croissante) et le rayon d’action des antennes (float). Cette fonction renvoie une liste d’objets de type ayant ce rayon d’action et couvrant toutes les maisons de la rue.

def strategie\_1(maisons, rayon):

    '''Prend en paramètre une liste de maisons et le rayon

       d'action des antennes et renvoie une liste d'antennes

    '''

    antennes = [Antenne(maisons[0].get\_pos\_maison(), rayon)]

    for m in maisons[1:]:

        if not antennes[-1].couvre(m):

            antennes.append(Antenne(m.get\_pos\_maison(), rayon))

    return antennes

Pour rappel :

* correspond aux éléments de tab à partir de l’indice 1 jusqu’à la fin de la liste ;
* correspond au dernier élément de la liste tab.

1. Indiquer ce que renvoie cette suite d’instructions après exécution.

>>> maisons = creation\_rue([0, 2, 3, 4, 5, 7, 9, 10.5, 11.5])

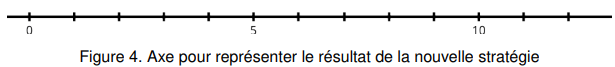
>>> antennes = strategie\_1(maisons, 2)

>>> print([a.get\_pos\_antenne() for a in antennes])

Une amélioration est possible et la société qui pose les antennes souhaite implémenter l’algorithme suivant :

* considérer les maisons dans l’ordre des abscisses croissantes ;
* dès qu’une maison n’est pas couverte, placer une antenne à la plus grande abscisse telle qu’elle couvre cette maison. Par exemple, si la maison d’abscisse 5 est la première maison non couverte, alors, on placera l’antenne en 5 + 𝑟 si 𝑟 est le rayon d’action de l’antenne.

1. On considère la rue composée des maisons situées aux abscisses . Recopier et compléter le schéma ci-dessous en indiquant l’emplacement des antennes selon cette nouvelle stratégie. On suppose que le rayon d’action est toujours 2.



1. Cet algorithme étant a priori plus économe en antennes, proposer une fonction , sur le modèle de qui implémente cette nouvelle stratégie.
2. Comparer le coût en nombre d’opérations des deux stratégies en fonction du nombre n de maisons dans la rue. On admet que le coût de la fonction est constant.

# Exercice 2 (6 points)

Cet exercice porte sur l’architecture matérielle, les réseaux, les routeurs et les protocoles de routage.

On considère un réseau local N1 constitué de trois ordinateurs M1, M2, M3 et dont les adresses IP sont les suivantes :

* M1 : 192.168.1.1/24 ;
* M2 : 192.168.1.2/24 ;
* M3 : 192.168.2.3/24.

On rappelle que le “/24” situé à la suite de l’adresse IP de M1 signifie que l’adresse réseau du réseau local N1 est 192.168.1.0.

Depuis l’ordinateur M1, un utilisateur exécute la commande ping vers l’ordinateur M3 comme suit :

util@M1 ~ % ping 192.168.2.3

PING 192.168.2.3 (192.168.2.3): 56 data bytes

Hôte inaccessible

1. Expliquer le résultat obtenu lors de l’utilisation de la commande ping (on part du principe que la connexion physique entre les machines est fonctionnelle).

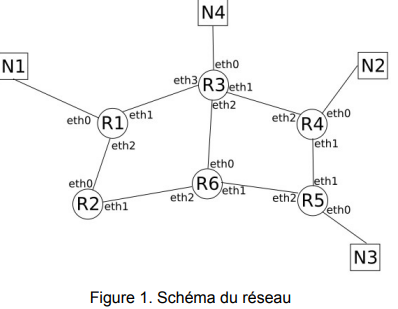
On ajoute un routeur R1 au réseau N1 :

“Un routeur moderne se présente comme un boîtier regroupant carte mère, microprocesseur, ROM, RAM ainsi que les ressources réseaux nécessaires (Wi-Fi, Ethernet…). On peut donc le voir comme un ordinateur minimal dédié, dont le système d’exploitation peut être un Linux allégé. De même, tout ordinateur disposant des interfaces adéquates (au minimum deux, souvent Ethernet) peut faire office de routeur s’il est correctement configuré (certaines distributions Linux minimales spécialisent la machine dans cette fonction).”

Source : Wikipédia, article “Routeur”

1. Définir l’acronyme RAM
2. Expliquer le terme LINUX
3. Expliquer pourquoi il est nécessaire d’avoir “au minimum deux“ interfaces réseau dans un routeur.

Le réseau N1 est maintenant relié à d’autres réseaux locaux (N2, N3, N4) par l’intermédiaire d’une série de routeurs (R1, R2, R3, R4, R5, R6) :



1. Attribuer une adresse IP valide à l’interface eth0 du routeur R1 sachant que l’adresse réseau du réseau N1 est 192.168.1.0.

Dans un premier temps, on utilise le protocole de routage RIP (Routing Information Protocol). On rappelle que dans ce protocole, la métrique de la table de routage correspond au nombre de routeurs à traverser pour atteindre la destination.

La table de routage du routeur R1 est donnée dans le tableau suivant :

|  |  |  |
| --- | --- | --- |
| TABLE DE ROUTAGE DU ROUTEUR R1 | | |
| Destination | Interface de sortie | Métrique |
| N1 | eth0 | 0 |
| N2 | eth1 | 2 |
| N2 | eth2 | 4 |
| N3 | eth1 | 3 |
| N3 | eth2 | 3 |
| N4 | eth1 | 1 |
| N4 | eth2 | 3 |

1. Déterminer le chemin parcouru par un paquet de données pour aller d’une machine appartenant au réseau N1 à une machine appartenant au réseau N2.

Le routeur R3 tombe en panne. Après quelques minutes, la table de routage de R1 est modifiée afin de tenir compte de cette panne.

1. Dresser la table de routage du routeur R1 suite à la panne du routeur R3.

Le routeur R3 est de nouveau fonctionnel. Dans la suite de cet exercice, on utilise le protocole de routage OSPF (Open Shortest Path First). On rappelle que dans ce protocole, la métrique de la table de routage correspond à la somme des coûts :

coût = (où d est la bande passante d’une liaison en bit/s).

Le réseau est constitué de 3 types de liaison de communication :

* Fibre avec un débit de 1 Gbit/s ;
* Fast-Ethernet avec un débit de 100 Mbit/s ;
* Ethernet avec un débit de 10 Mbit/s.

1. Calculer le coût de chacune de ces liaisons de communication.

La table de routage du routeur R1 est donnée dans le tableau suivant :

|  |  |  |
| --- | --- | --- |
| TABLE DE ROUTAGE DU ROUTEUR R1 | | |
| Destination | Interface de sortie | Métrique |
| N1 | eth0 | 0 |
| N2 | eth1 | 10.1 |
| N2 | eth2 | 1.3 |
| N3 | eth1 | 11.3 |
| N3 | eth2 | 0.3 |
| N4 | eth1 | 10 |
| N4 | eth2 | 1.2 |

D’autre part, le type des différentes liaisons inter-routeurs sont les suivantes :

* R1 - R2 : Fibre ;
* R1 - R3 : Ethernet ;
* R2 - R6 : INCONNU ;
* R3 - R6 : Fast-Ethernet ;
* R3 - R4 : Fibre ;
* R4 - R5 : Fast-Ethernet ;
* R5 - R6 : Fibre.

1. Déduire de la table de routage de R1 et du schéma du réseau le type de la liaison inter-routeur R2 - R6.$

Des travaux d’amélioration ont été réalisés sur ce réseau : la liaison inter-routeur R1- R3 est désormais de type Fibre.

1. Modifier la table de routage de R1 en tenant compte de cette amélioration.

On ajoute un réseau local N5 et un routeur R7 au réseau étudié ci-dessus. Le routeur R7 possède trois interfaces réseaux eth0, eth1 et eth2. eth0 est directement relié au réseau local N5. eth1 et eth2 sont reliés à d’autres routeurs (ces liaisons inter-routeur sont de type Fibre).

Les deux tableaux suivants présentent un extrait des tables de routage des routeurs R1 et R3 :

|  |  |  |
| --- | --- | --- |
| TABLE DE ROUTAGE DU ROUTEUR R1 | | |
| Destination | Interface de sortie | Métrique |
| ... | … | … |
| N5 | eth1 | 1.2 |
| N5 | eth2 | 0.2 |

|  |  |  |
| --- | --- | --- |
| TABLE DE ROUTAGE DU ROUTEUR R3 | | |
| Destination | Interface de sortie | Métrique |
| ... | … | … |
| N5 | eth1 | 1.3 |
| N5 | eth2 | 1.1 |
| N5 | eth3 | 0.3 |

1. Recopier et compléter le schéma du réseau (Figure. 1) en ajoutant le routeur R7 et le réseau local N5.

# Exercice 3 (8 points)

Cet exercice porte sur la programmation Python (dictionnaire), les bases de données relationnelles et les requêtes SQL.

Le Tour de France est une course cycliste qui se déroule chaque année. Chaque jour, les coureurs s’affrontent pour remporter l’étape du jour, ce qui détermine un classement d’étape. Le coureur avec le temps cumulé le plus bas sur l’ensemble des étapes mène le classement général. Chaque participant est repéré par un dossard et appartient à une équipe. En 2023, 22 équipes de 8 coureurs, soit 176 cyclistes ont pris le départ du tour.

## PARTIE A

Dans cette partie, nous allons utiliser trois dictionnaires :

Le premier, appelé , a pour clés les noms complets des coureurs et pour valeurs les numéros de dossard correspondants.

Le deuxième, appelé , utilise les numéros de dossard comme clés et contient une liste des temps d’arrivée de chaque étape en seconde.

Le troisième, appelé , utilise également les numéros de dossard comme clés et indique le classement général mis à jour à la fin de chaque nouvelle étape.

Par exemple, à la fin de la quatrième étape, voilà les trois premiers éléments de ces dictionnaires :

participants = {"VINGEGAARD Jonas": 1, "BENOOT Tiesj": 2, "KELDERMAN Wilco": 3, ...}

temps\_etapes = {1: [15781, 17199, 16995, 15928], 2: [15960, 17199, 16995, 15928], ...}

classement\_general = {1: 6, 2: 30, 3: 13, ...}

1. En utilisant ces dictionnaires, écrire une instruction permettant d’obtenir :

* le numéro de dossard de PHILIPSEN Jasper ;
* le classement général de PHILIPSEN Jasper ;
* le temps, en seconde, mis par le cycliste PINOT Thibaut pour courir la quatrième étape.

1. Écrire une fonction qui a pour paramètre le numéro d’un dossard d et qui renvoie le temps total en seconde mis par ce coureur depuis le départ du tour de France.
2. Le dictionnaire étant remis à jour après la fin d’une étape, recopier et compléter les lignes 8, 9 et 14 du programme suivant afin que le dictionnaire classement\_general soit aussi mis à jour.

for numero\_dossard in temps\_etapes:

    element = (numero\_dossard, calcul\_temps\_total(numero\_dossard))

    classement.append(element)

    pos = len(classement) - 2

    while pos >= 0 and element[...] < classement[pos][...]:

        classement[pos + 1] = ...

        pos = pos - 1

        classement[pos + 1] = element

for i in range(len(classement)):

    classement\_general[...] = i + 1

On suppose qu’on dispose d’un tableau tableau\_temps composé de tuples contenant le numéro du dossard, le nom, et le temps total en seconde de chaque coureur, trié par ordre croissant de temps. On donne ci-dessous un aperçu du début du tableau :

tableau\_temps = [(1, "VINGEGAARD Jonas", 65903),

 (3, "KELDERMAN Wilco", 65987),

 (2, "BENOOT Tiesj", 66082),

 ...]

On souhaite créer une variable Python tableau\_final de type liste de listes :

[[1, "VINGEGAARD Jonas", 65903],

[3, "KELDERMAN Wilco", 84],

[2, "BENOOT Tiesj", 179]]

...

Pour le premier, la troisième valeur de la première liste est le temps total mis par le vainqueur. Pour les autres coureurs, la troisième valeur des autres listes est l’écart de temps mis avec le premier.

Par exemple : 84 = 65987 - 65903 et 179 = 66082 - 65903.

1. Recopier et compléter le programme pour qu’il en soit ainsi

tableau\_final = []

difference\_temps = 0

premier = True

for ligne in tableau\_temps:

    coureur = [ligne[0]]

    coureur.append(ligne[1])

    if premier:

        temps\_premier = ligne[2]

        coureur.append(temps\_premier)

        premier = False

    else:

        difference\_temps = ligne[2] - ...

        coureur.append(...)

        tableau\_final.append(...)

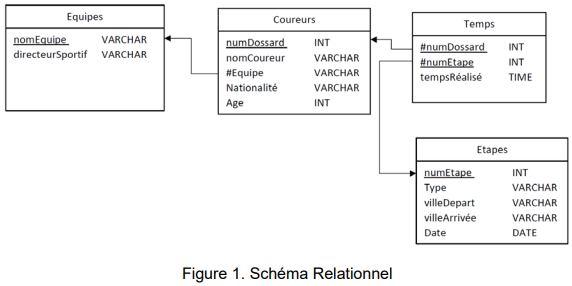
## PARTIE B

Dans cet exercice, on pourra utiliser les mots clés suivants du langage SQL

SELECT, FROM, WHERE, JOIN, ON, INSERT INTO, VALUES, MIN, MAX, OR, AND et ORDER BY.

Si est un des attributs d’une relation, les fonctions d’agrégation , renvoient, respectivement, la plus petite, la plus grande valeur et la somme des valeurs des attributs sélectionnés.

On considère la base de données du tour de France 2023 dont le schéma relationnel est donné ci-dessous :



Dans ce schéma, les clés primaires sont soulignées et les clés étrangères sont précédées du symbole #.

1. Expliquer pourquoi, dans la relation Temps, il est nécessaire de prendre le couple (numDossard, NumEtape) comme clé primaire.
2. Expliquer ce que renvoie la requête SQL suivante :

SELECT nomCoureur

FROM Coureurs

WHERE Equipe = 'Cofidis';

1. Écrire une requête SQL permettant d’obtenir les dates de toutes les étapes de type 'contre-la-montre' du tour de France 2023.
2. Écrire une requête SQL permettant d’obtenir le nom du directeur sportif du coureur BARDET Romain.
3. Écrire une requête SQL donnant le temps total en course mis par BARDET Romain depuis le départ du tour de France 2023.