Épreuve disciplinaire appliquée

Éléments de correction

Préambule : cette épreuve est constituée de deux problèmes indépendants. Les réponses aux questions devront être précises et rédigées avec soin.

Problème 1 - Gestion des données

1 Introduction à la problématique des données

Vous souhaitez aborder la notion de données et l'importance que les données ont pris dans notre société avec vos élèves de SNT en seconde.

1. Citer un exemple que vous pourriez utiliser devant les élèves pour illustrer l'importance de l'exploitation des données dans leur vie quotidienne. Vous justifierez brièvement votre choix.

   Éléments de réponse

   Voici des exemples possibles :
   - Récupération de leurs données d'écoute/visionnage pour leur proposer du contenu ciblé (musique/vidéo/réseaux sociaux) (l'intérêt consiste à montrer que se baser sur les données a des avantages -ici avoir du contenu qui nous intéresse- mais aussi des inconvénients - les informations proposées sont basées sur ce que l'on a précédemment consulté, ce qui peut induire des biais de confirmation-, et donc d'apporter un propos nuancé).
   - Données en temps réel : circulation routière en temps réel (exemple qui devrait parler aux élèves car leurs parents utilisent des gps même si eux ne conduisent pas encore).

   L'important est que les exemples parlent vraiment aux élèves.
2. Expliquer, en quelques lignes et comme vous l'expliqueriez à vos élèves, pourquoi le stockage de données sur le « cloud » (donc dans les datacenters) a bien un impact négatif sur l'environnement. Vous donnerez deux impacts différents sur l'environnement.

   Éléments de réponse

   Les datacenters qui stockent les données du Cloud fonctionnent bien sûr avec de l'électricité, mais doivent aussi, pour pouvoir fonctionner correctement, être refroidis, ce qui a un coût énergétique non négligeable. De plus, les machines sur lesquelles les données sont stockées doivent être fabriquées et pour cela on utilise notamment des métaux rares (dont l'extraction est très polluante). Il faut aussi transférer les informations depuis le datacenter vers les appareils et vice-versa ce qui a aussi un coût énergétique (et nécessite aussi une infrastructure de communication).
3. Donner deux conseils concrets que vous pourriez donner à vos élèves pour limiter facilement l'impact de leurs données sur l'environnement.

   Éléments de réponse

   Il y a bien sûr beaucoup de possibilités et il s'agit en général de réduire la quantité de données (stockées et/ou téléchargées). Voici quelques exemples :
   - Supprimer les mails qui ne sont plus utiles dans leur boîte
   - Supprimer les fichiers inutiles dans leur Cloud
   - Ne pas envoyer une vidéo quand un simple SMS suffit, ou favoriser les formats de compression récents ou des définitions inférieurs (éviter le HD ou UltraHD)
   - Passer par un câble Ethernet pour télécharger plutôt que le Wi-Fi

- Sauvegarder en local les fichiers/vidéos dont on aura besoin souvent plutôt que de les retélécharger à chaque fois

4. Expliquer en quoi disposer de données massives (appelées aussi Big Data) peut être un avantage dans certains enjeux de société. Illustrer votre réponse avec un exemple que vous pourriez proposer à vos élèves.

Éléments de réponse

Voici quelques exemples possibles. En intelligence artificielle, on peut être amené à manipuler des grands volumes de données (jeu d'entraînement et de test) pour la construction de méthodes de prédiction. C'est à l'origine de progrès dans de nombreux domaines scientifiques comme, par exemple, la santé, les prévisions météo. Disposer de données massives permet aussi l'optimisation dans des domaines très variés (boutique e-commerce, horaires de trains). À New-York, le Big Data permet par exemple de prévoir les incendies. C'est une science qui se développe et beaucoup d'usages ne sont pas encore connus.

5. Citer un danger (autre que l'impact sur l'environnement) qui est apparu ou peut apparaître avec les données massives. Vous expliquerez, comme vous l'expliqueriez à vos élèves, ce danger en quelques lignes.

Éléments de réponse

Voici quelques dangers possibles :

- Le risque de surveillance de masse et l'atteinte à la vie privée. L'analyse du trafic Internet ou encore la géolocalisation en temps réel via des systèmes de reconnaissance faciale, sont deux dispositifs présentant ces risques, et ont déjà été utilisés dans certains pays à des fins de répression politique.
- La sécurité des données. En effet, la mutualisation des données, les informations personnelles pouvant en être tirées ainsi que l'exploitation commerciale en découlant, en font une cible privilégiée pour des individus ou organismes malintentionnés (tentatives d'arnaques, démarchages agressifs).

2 Utilisation des fichiers csv

Vous voulez maintenant traiter la notion de données structurées toujours avec vos élèves de seconde.

6. Définir la notion de descripteur d'objet.

Éléments de réponse

Descripteur : caractéristique propre à un objet permettant de le décrire au sein d'une collection

7. Proposer une activité débranchée que vous pourriez réaliser en classe pour leur faire comprendre la notion de descripteur d'objet. Cette activité devra durer entre 30 min et 1h. On attend une description détaillée du déroulement de l'activité et de ses attendus.

Éléments de réponse

Proposition d'activité débranchée : les élèves choisissent un objet/une personne (un camarade, une célébrité, un film, un livre,...) et les autres élèves doivent être capables de retrouver l'objet/personne grâce à la description faite par leurs camarades. Au fur et à mesure on complique les choses en empêchant l'utilisation de certains descripteurs (le nom pour les personnes par exemple). On fait alors émerger le terme de descripteurs d'une donnée. On peut alors les sensibiliser aux différents types de données (texte, nombre,...) et les faire réfléchir aux descripteurs qui permettent à coup sûr d'identifier la personne/l'objet. On peut ensuite remettre en application sur un autre exemple.

Vous abordez ensuite le traitement de ces données structurées. Voici l'extrait du programme de seconde SNT sur ce point :

Contenus	Capacités attendues
Traitement de données structurées	Réaliser des opérations de recherche, filtre, tri ou calcul sur une ou plusieurs tables

Pour cela, vous décidez de les faire travailler sur le fichier csv suivant : « Donnees_capteurs_22_09_21.csv » (cf Annexe A). Ce fichier contient toutes les mesures réalisées par les capteurs d'une maison domotisée le 22/09/2021. Chaque capteur envoie ses mesures toutes les 5 minutes. Ce sont des capteurs polyvalents placés sur les ouvertures d'une maison (fenêtres/portes) qui permettent de détecter l'ouverture ou non mais aussi le mouvement et la température. Les différentes colonnes disponibles dans ce fichier sont :

- Identifiant_capteur (entier) : identifiant unique du capteur.

- Piece (texte) : pièce dans laquelle se trouve le capteur.
- Num_mesure (entier) : numéro de la mesure de la journée pour le capteur (1 à minuit et s'incrémente à chaque mesure jusqu'à la fin de la journée).
- Ouverture (booléen) : vaut True si la porte/fenêtre est ouverte et False sinon.
- Presence (booléen) : vaut True si le capteur détecte un mouvement et False sinon.
- Temperature (flottant) : indique la température mesurée à 0.1°C près.

Le fichier csv est pour l'instant ouvert par les élèves dans un tableur comme Calc/Excel.

8. Donner 3 questions que vous pourriez poser aux élèves pour les faire travailler, sur ordinateur, sur le traitement de données structurées, le but étant ici de leur faire découvrir les traitements possibles. Vous expliquerez à chaque fois brièvement l'intérêt de la question pour l'acquisition de compétence des élèves. Chaque question devra faire travailler les élèves sur une opération différente.

Éléments de réponse

Il faut que les questions fassent travailler sur la recherche, le filtre, le tri ou le calcul. Quelques exemples :
- N'afficher que les données des capteurs de la cuisine. (filtre)
- Afficher les données en les triant par l'identifiant de capteur. (tri)
- N'afficher que les données du capteur n°4 dans l'ordre décroissant des mesures (de la fin de journée au début de la journée). (filtre + tri)
- Calculer la température moyenne de la cuisine entre 8h et 20h. (calcul-moyenne)
- Y a-t-il eu quelqu'un dans le garage lors de cette journée ? (recherche)
- Calculer la température maximale/minimale d'une pièce (calcul-max/min)
- Déterminer pendant combien de temps une porte (identifiée par son capteur) est restée ouverte sur la journée. (filtre + calcul-nombre de fois)

Vous souhaitez maintenant les faire travailler sur le traitement du fichier à l'aide de Python en utilisant la bibliothèque pandas. Vous trouverez en annexe une description des fonctions utiles de pandas (cf Annexe B).

Pour une exploitation du fichier « Donnees_capteurs_22_09_21.csv » avec la bibliothèque pandas, on s'est assuré que les nombres réels sont désormais écrits avec des points et non des virgules.

Le but de votre exercice est que les élèves puissent répondre de manière automatisée grâce à Python aux questions suivantes (pour le jour considéré) :
- Donner la température minimale de la cuisine quand le capteur 3 indique la présence de quelqu'un.
- Afficher toutes les températures de la maison triées de manière croissante.

9. Écrire, tel qu'un élève pourrait le faire, le programme permettant de répondre à ces questions à l'aide de la bibliothèque pandas.

Éléments de réponse

[

```
import pandas
donnees=pandas.read_csv("Donnes_capteurs_22_09_21.csv",sep=';')
print(donnees.loc[(donnees['Piece']=='Cuisine') & (donnees['Presence']==True) ,'
    Temperature'].min())
print(donnees.loc['Temperature'].sort_values( ascending=True))
```

]

10. Décrire la structure d'une activité de 1h30 sur machine permettant à vos élèves, qui n'ont jamais utilisé pandas, d'être capables de répondre à ces questions. On ne demande pas ici de rédiger complètement le document qui serait fourni aux élèves mais d'indiquer clairement l'organisation de l'activité en précisant notamment les questions posées aux élèves.

Éléments de réponse

Voici une description succincte d'une organisation possible de l'activité :
- Leur faire construire progressivement des éléments de réponses aux questions (pas exemple, afficher les données puis sélectionner une colonne, mettre un critère, en mettre plusieurs,...).
- Les faire travailler sur la sélection sur critère, puis sur le tri des valeurs (d'abord simplement sur les colonnes puis sur une sélection de données).

Les élèves pourrait d'abord travailler sur des exemples fournis, puis être amenés à écrire eux-mêmes les programmes.

3 Traitement de données sous forme de tables

Vous travaillez sur le traitement de données en tables avec vos élèves de première. Pour travailler sur les données en tables, vous avez choisi de travailler avec des tableaux de p-uplets. Avec les élèves, les p-uplets seront codés, en Python, à l'aide de dictionnaires. Dans cet objectif, vous préparez un cours sur les dictionnaires.

11. Présenter, tel que vous le feriez avec vos élèves, la différence entre le type dictionnaire et le type liste.

    Éléments de réponse

    Les dictionnaires et les listes permettent de rassembler des éléments. Les éléments d'une liste sont ordonnés selon un numéro qu'on appelle l'indice de l'élément. On accède à un élément en utilisant cet indice. Dans un dictionnaire en Python, les éléments sont identifiés par une clé. On peut faire l'analogie avec un dictionnaire de français où on accède à une définition avec un mot. Contrairement aux listes qui sont délimitées par des crochets, on utilise des accolades pour les dictionnaires.

12. Expliquer brièvement comment les dictionnaires sont implémentés en Python.

    Éléments de réponse

    En Python, les dictionnaires sont implémentés sous forme de tables de hachage. Chaque entrée du tableau est en fait une combinaison de trois valeurs : <valeur-hachage, clé, valeur>. Lors de l'ajout d'une entrée à la table, le hachage de la clé donne un emplacement. Si cet emplacement est vide, l'entrée est ajoutée à l'emplacement. Si ce n'est pas le cas (et qu'il s'agit bien d'une nouvelle clé à insérer), il recherche un emplacement vide. Depuis Python 3.7 (officiellement 3.6 officieusement) les dictionnaires sont désormais classés selon l'ordre d'insertion..

13. Citer un exemple d'utilisation de dictionnaire pour illustrer votre cours sur les dictionnaires. Vous justifierez votre choix en expliquant notamment pourquoi l'exemple est pertinent pour ce cours de première. Qu'est-ce qui pourrait être un mauvais exemple d'un point de vue pédagogique et pourquoi ?

    Éléments de réponse

    Quelques exemples :

    - Un « annuaire » contenant des noms et des numéros de téléphone.
    - Un dictionnaire français-anglais.

    L'exemple est pertinent s'il est parlant pour les élèves, simple (on est au début du cours) et montre bien l'intérêt du dictionnaire par rapport aux listes. Il doit aussi permettre d'illustrer qu'ici l'ordre des couples clé-valeur n'a pas d'importance (données pas ordonnées). Un mauvais exemple peut être un exemple qui associe à un ensemble d'entiers consécutifs, un texte (ou un nombre). Par exemple un relevé météo de températures relevées quotidiennement sur une année sera directement exploitable en étant placé dans une liste de 365 éléments. L'utilisation d'un dictionnaire dans ce cas n'est pas nécessaire.

Vous décidez de travailler avec vos élèves à partir de deux fichiers csv présentés en annexe pour démarrer le traitement des données en tables (annexe C). Pour que les élèves comprennent bien les mécanismes en jeu, vous n'utilisez pas ici la bibliothèque Python pandas.
Vous vous êtes inspirés de la situation sanitaire actuelle pour créer deux fichiers (qui ne sont bien-sûr que des fichiers exemples) : l'un correspondant à la liste des personnes testées positives à la Covid19 et l'autre à la liste des personnes ayant mangé au restaurant. Ces fichiers sont supposés être stockés par l'assurance maladie, le second servant à contacter les personnes contacts à risques.

Le premier fichier nommé « positifs.csv » (cf Annexe C.1) contient les informations suivantes :

- le numéro de sécurité social du contaminé (descripteur : num_SS ; type : entier),
- son âge (descripteur : age ; type : entier),
- le jour du test positif (descripteur : jour_test ; type : entier),
- le mois du test positif (descripteur : mois_test ; type : entier),
- son sexe (descripteur : sexe ; type : caractère),
- s'il a été vacciné ou non (descripteur : vaccine ; type : booléen).

Le deuxième fichier nommé « restaurants.csv » (cf Annexe C.2) contient les informations suivantes :

- l'identifiant de l'enregistrement (descripteur : id_enregistrement ; type : entier),
- l'identifiant du restaurant (descripteur : id_restaurant ; type : entier),
- le jour du repas (descripteur : jour_repas ; type : entier),
- le mois du repas (descripteur : mois_repas ; type : entier),

- la tranche horaire du repas (descripteur : tranche_horaire ; type : entier),
- le numéro de la table (descripteur : num_table ; type : entier),
- le numéro de sécurité social du client (descripteur : num_SS ; type : entier).

Pour cet exercice, on ne considèrera que 2 tranches horaires (la 1 de 12h à 14h et la 2 de 19h à 21h) et on considérera qu'une même personne ne peut pas être testée deux fois positive à la Covid19.

14. Écrire le programme Python commenté qui permettrait à vos élèves de récupérer sous forme de liste de dictionnaires, nommée **positifs**, les informations contenues dans le fichier « positifs.csv ». Chaque élément de la liste est un dictionnaire comprenant les valeurs d'une ligne de la table et dont les clés sont les descripteurs de la table. On utilisera pour cela notamment la fonction readlines.

Éléments de réponse

[

```
f=open('positifs.csv','r')
Ltexte=f.readlines()
f.close()

champs=Ltexte[0].strip().split(";")

positifs=[]
for ligne in Ltexte[1:]:
    enregistrement={}
    donnees=ligne.strip().split(";")
    for i in range(0,4):
            enregistrement[champs[i]]=int(donnees[i])
    enregistrement[champs[4]]=donnees[4]
    enregistrement[champs[5]]=eval(donnees[5])
    positifs.append(enregistrement)
```

]

15. Quels sont pour vous les points importants à aborder avec vos élèves et/ou ceux qui risquent de leur poser problème lorsque vous abordez pour la première fois le programme de la question 14 avec vos élèves ?

Éléments de réponse

- Les sensibiliser à la fonction readlines et au fait que sa valeur de retour est une liste dont chaque élément est un texte.
- Les sensibiliser à la fonction split qui ne s'utilise que sur un texte (et non pas sur une liste) mais qui renvoie une liste.
- Attention aux types manipulés. On part toujours d'un texte mais suivant les données, il faut changer de type pour pouvoir travailler dessus ensuite.
- Le comptage démarre à la valeur 0. Des élèves de 1ère peuvent ne pas être familiarisés à cet usage.
- Faire attention aux en-têtes qui sont utilisés pour avoir les clés mais qui ne doivent pas être traités dans la boucle for.

Vous souhaitez faire travailler vos élèves sur la recherche, dans une table, des lignes vérifiant des critères exprimés en logique propositionnelle en utilisant les fichiers décrits ci-dessus. On supposera, pour la suite de l'exercice, que les élèves ont implémenté le programme Python retournant une liste de dictionnaires **restaurants** contenant les informations contenues dans le fichier « restaurants.csv ».

16. Vous rédigez maintenant une partie de cet exercice.
    (a) Proposer 3 questions de difficulté croissante (dont au moins une nécessitant la fusion des tables) où les élèves doivent rédiger des programmes en Python. On suppose ici que les listes de dictionnaires **positifs** et **restaurants** sont déjà construites.

Éléments de réponse

Voici un exemple de 3 questions :

i. Donner la liste des numéros de sécurité sociale des personnes testées positives et qui sont vaccinées.

ii. Donner la liste des numéros de sécurité sociale des personnes ayant mangé le 30/06 ou le 01/07 .

iii. Donner la liste des identifiants des restaurants où une personne positive a mangé (la liste pourra contenir plusieurs fois le même identifiant).

(b) Proposer pour chaque question une correction. Décrire brièvement les points sur lesquels vous serez vigilant lors de la correction de ces questions.

Éléments de réponse

[

1.

```
Lpositifs=[]
for pos in positifs:
    if pos['vaccine']==True:
        Lpositifs.append(pos['num_SS'])
print(Lpositifs)
```

2.

```
Lnum=[]
for enr in restaurants:
    if (enr['jour_repas']==30 and enr['mois_repas']==6) or (enr['jour_repas']==1
        and enr['mois_repas']==7):
        Lnum.append(enr['num_SS'])
print(Lnum)
```

3. Ici on crée un liste de dictionnaires qui contient la fusion et on travaille dessus. Ce n'est pas obligatoire.

```
fusion=[]
for p in positifs:
    for enr in restaurants:
        if enr['num_SS']==p['num_SS']:
            dict={}
            for cle in p.keys():
                dict[cle]=p[cle]
            for cle in enr.keys():
                dict[cle]=enr[cle]
            fusion.append(dict)

for elt in fusion:
    print(elt['id_restaurant'])
```

]

Éléments de réponse

Une syntaxe spécifique Python peut aussi être proposée dans ces 3 questions.

1.

```
Lpositifs=[pos['num_SS'] for pos in positifs if pos['vaccine']]
```

2.

```
Lnum=[enr['num_SS'] for enr in restaurants
      if (enr['jour_repas'] == 30 and enr['mois_repas'] == 6)
      or (enr['jour_repas'] == 1 and enr['mois_repas'] == 7)]
```

3.

```
Lfusion = [enr['id_restaurant'] for enr in restaurants if enr['num_SS'] in
       [pos['num_SS'] for pos in positifs]]
```

4 Utilisation des bases de données

Dans cette partie, vous travaillez sur les bases de données avec vos élèves de terminale.

17. Expliquer tel que vous le feriez avec vos élèves l'intérêt des bases de données par rapport à ce que les élèves ont fait en première (avec les tables sous forme de fichiers csv).

Éléments de réponse

Une base de données permet de mettre des données à la disposition d'utilisateurs pour une consultation, une saisie ou bien une mise à jour, tout en s'assurant des droits accordés à ces derniers. Plusieurs avantages peuvent être indiqués : i) La gestion multi-utilisateurs ; ii) La robustesse des données (contrôle des transactions et persistance en cas de panne) ; iii) L'existence d'outils de sauvegarde et de synchronisation des bases de données.

Voici un extrait du programme de NSI de terminale.

Contenus	Capacités attendues	Commentaires
Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données.	Identifier les composants d'une requête. Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : SELECT, FROM, WHERE, JOIN. Construire des requêtes d'insertion et de mise à jour à l'aide de : UPDATE, INSERT, DELETE.	On peut utiliser DISTINCT, ORDER BY ou les fonctions d'agrégation sans utiliser les clauses GROUP BY et HAVING.

Vous avez choisi, pour travailler sur le langage SQL, un exercice avec une base de données simplifiée qui stocke les données d'un service proposant des films en streaming. La base de données contient les tables suivantes :

Films ={Id, Titre, Annee, Pays}
Clients= {Id, Nom, Prenom, AdresseMail, TypeAbonnement}
Visionnage={Id, IdClient, IdFilm, Date, Heure, TempsVisionnage}
Facturation={Id, IdClient, Montant, Date, Statut}

Un extrait de cette base de données est fournie dans l'annexe D. Le temps de visionnage est stocké en minutes. Il existe deux types d'abonnement : le "basic" et le "premium". Les factures peuvent être "Payée" ou "En attente".

18. Voici l'exercice que vous avez donné aux élèves pour travailler sur les requêtes en SQL :
    (a) Écrire la requête SQL permettant d'ajouter le film suivant à la base de données : « Wicked » un film réalisé aux USA et sorti en 2021. Son identifiant est 20356.
    (b) Écrire la requête SQL permettant d'indiquer que la facture d'identifiant 2564 a été payée.
    (c) Écrire la requête SQL permettant d'obtenir la liste des identifiants de facturation qui sont en attente et qui ont un montant strictement supérieur à 5 euros.
    (d) Écrire la requête SQL permettant d'obtenir la liste, par ordre alphabétique, des pays d'origine des films de la plateforme de l'année 2015. Chaque pays ne sera bien sûr indiqué qu'une seule fois.
    (e) Écrire la requête SQL permettant d'obtenir la liste des noms, prénoms et adresses mails des clients ayant une facturation en attente.
    (f) Écrire la requête permettant de déterminer le temps moyen de visionnage du film Titanic.
    (g) Écrire la requête permettant de donner la liste des identifiants des clients qui ont regardé Titanic plus longtemps que la moyenne des utilisateurs.

    Donner les réponses que vous pourriez attendre pour chacune de ces questions.

Éléments de réponse

a. INSERT INTO Films VALUES (20356,"Wicked",2021,"USA") ;
b. UPDATE Facturation SET Statut="Payée" WHERE Id = "2564" ;
c. SELECT Id FROM Facturation WHERE Statut = "En attente" AND Montant>5 ;
d. SELECT DISTINCT Pays FROM Films WHERE Annee=2015 ORDER BY Pays ;
e. SELECT Nom,Prenom,AdresseMail FROM Clients JOIN Facturation ON Clients.Id=Facturation.IdClient WHERE Statut= "En attente " ;
f. SELECT AVG(TempsVisionnage) FROM Visionnage JOIN Films ON Visionnage.IdFilm=Films.Id WHERE Titre= "Titanic" ;
g. SELECT IdClient FROM Visionnage WHERE TempsVisionnage > SELECT AVG(TempsVisionnage) FROM Visionnage JOIN Films ON Visionnage.IdFilm=Films.Id WHERE Titre="Titanic"
(Autre alternative - compatible avec SQL92 : SELECT IdClient FROM Visionnage JOIN (SELECT AVG(TempsVisionnage) As TempsMoy FROM Visionnage JOIN Films ON Visionnage.IdFilm=Films.Id WHERE Titre="Titanic") WHERE TempsVisionnage >TempsTempsMoy)

19. Des élèves peinent à rédiger la requête d'interrogation de données de la question 18(e). Quelle aide leur proposez-vous ?

Éléments de réponse

On peut les faire s'interroger progressivement :
- Dans quelle table se trouve l'information sur la facturation ? Contient-elle les noms, prénoms et mails des clients ? Où sont stockées ces informations ?
- Maintenant qu'on sait qu'il faut fusionner les deux tables, quelle information permet de faire la fusion ?

Puis leur faire écrire la requête morceau par morceau :
- faire écrire une requête avec seulement la jointure (en affichant la table obtenue),
- faire la sélection des bonnes informations,
- faire ajouter la condition.

20. Donner un barème pour la correction de la question 18(g). Vous détaillerez les différentes compétences qu'il vous semble important d'évaluer.

Éléments de réponse

Voici la proposition d'un barème :
- Avoir compris qu'il faut une requête pour calculer le temps moyen (1 pt)
- Jointure entre visionnage et films correctement réalisée (1 pt)
- Valeur moyenne correctement obtenue dans la sous-requête (1 pt)
- Données correctement sélectionnées dans la requête principale (1 pt)

21. Proposer, à l'aide de cette base de données, 3 questions qui vous permettraient de valider les compétences de vos élèves sur les requêtes SQL lors d'une interrogation. Vous justifierez, pour chaque question, en quoi elle est pédagogiquement intéressante en indiquant notamment la (ou les) compétence(s) évaluée(s).

Éléments de réponse

Les questions doivent faire travailler des points variés du programme (insertion/mise à jour et recherche) et être d'une difficulté raisonnable.
Voici des questions possibles :
Question 1 : Donner les montants dépensés pour chaque IdClient.
SELECT IdClient, SUM(Montant) As Depense From Facturation GROUP BY IdClient
Cette question permet d'évaluer la compétence : Effectuer des opérations sur des données extraites.

Question 2 : Donner les adresse email des clients n'ayant rien visionné depuis 15 jours.
SELECT AdresseMail From Clients MINUS SELECT AdresseMail From Clients JOIN Visionnage ON Clients.Id = IdClient WHERE Date $\geqslant$ NOW() - 15
Cette question permet d'évaluer la compétence : Opérer sur les résultats de deux requêtes.

Question 3 : Quels sont les 5 films qui ont été les plus (en tout ou partie) regardés ?
SELECT Titre FROM Films JOIN (SELECT IdFilm, COUNT(Id) AS Total FROM Visionnage GROUP BY IdFilm ORDER BY Total DESC LIMIT 5) As Totaux ON Films.Id = Totaux.IdFilm
Cette question permet d'évaluer la compétence : Imbriquer des requêtes.

Problème 2 - Algorithmes de tri

Trier une collection de données avant de résoudre un problème algorithmique permet souvent d'élaborer des solutions beaucoup plus efficaces. Par exemple, on peut effectuer une recherche dichotomique au lieu d'effectuer un parcours linéaire dans une collection triée ou le tri d'un nuage de points permet de calculer efficacement l'enveloppe convexe de ce nuage de points (parcours de Graham).

De nombreux algorithmes de tri procèdent par comparaisons successives d'éléments entre eux. Pour évaluer l'efficacité de ces algorithmes et les comparer, on évalue souvent le nombre de comparaisons effectuées lors de leur exécution : on parle alors de **complexité en nombre de comparaisons**. Si l'on considère d'autres opérations élémentaires telles que les affectations, on dit que l'on évalue la **complexité temporelle**. Dans ce sujet, en réponse aux questions de complexité, on attend des ordres de grandeur en notation Landau (par exemple $O(n \log n)$ ou $O(n^2)$ si n est le nombre d'éléments à trier). Certains algorithmes de tri peuvent avoir des ordres de grandeur différents pour la complexité en nombre de comparaisons et la complexité temporelle. Nous garderons cela à l'esprit lors de l'analyse des algorithmes de tri de ce sujet. Enfin, on rappelle qu'un algorithme de tri par comparaisons correct nécessite au moins $n \log n$ comparaisons dans le pire cas. Si nécessaire, vous pourrez vous référer à cette borne considérée comme admise par la suite.

La dernière section de cette partie concerne des algorithmes de tri qui n'utilisent pas de comparaison, nous évaluerons alors seulement la complexité temporelle.

5 Tris naïfs

Dans cette section, deux algorithmes de tri par comparaisons du programme de première NSI sont considérés : le tri par insertion et le tri par sélection. Ces tris sont dits naïfs car leur complexité n'est pas optimale.

5.1 Tri par insertion

22. Quatre élèves de première NSI vous ont rendu leur implémentation du tri par insertion. Pour chaque production, rédiger une appréciation concise, donnant les erreurs éventuelles et les améliorations possibles afin que les élèves puissent se corriger par elles-mêmes.

```
def tri_eleve1(t) :
    n = len(t)
    for i in range(n) :
        for k in range(i,1,-1) :
            if t[k]<t[k-1] :
                t[k]=t[k-1]
                t[k-1]=t[k]
```

```
def insere_eleve2(t,k):
    if t[k]<t[k-1] :
        t[k],t[k-1]=t[k-1],t[k]
        insere_eleve2(t,k-1)

def tri_eleve2(t,i=0) :
    insere_eleve2(t,i)
    tri_eleve2(t,i+1)
```

```
def tri_eleve3(t):
    n = len(t)
    def insere_eleve3(i) :
        k = i
        while t[k]<t[k-1] :
            t[k],t[k-1]=t[k-1],t[k]
            k = k-1
    for i in range(n) :
        t = insere_eleve3(i)
```

```
def tri_eleve4(t) :
    n = len(t)
    i = 0
    while i < n :
        k = i
        while k > 0 or t[k]<t[k-1] :
            t[k],t[k-1]=t[k-1],t[k]
        i = i + 1
```

Éléments de réponse

Pour obtenir des productions d'élèves "correctes", voici ce qu'il faut modifier.

- `tri_eleve1` : il faut `range(i,0,-1)` sinon l'élément `t[0]` n'est jamais modifié ; l'échange de `t[k]` et `t[k-1]` ne fonctionne pas,
- `tri_eleve2` : il manque les deux cas de base,
- `tri_eleve3` : il manque la condition `k>0` dans la boucle `while` et `t = insere_eleve3(i)` provoque une erreur car la valeur de `insere_eleve3(i)` est `None`,
- `tri_eleve4` : il manque le décrément de `k` dans la boucle `while` et le `or` devrait être un `and`.

Les productions d'élèves corrigées sont données ci-dessous :

```
def tri_eleve1(t) :
    n = len(t)
    for i in range(n) :
        for k in range(i,0,-1) :
            if t[k]<t[k-1] :
                t[k],t[k-1]=t[k-1],t[k]
```

```
def insere_eleve2(t,k):
    if k>0 and t[k]<t[k-1] :
        t[k],t[k-1]=t[k-1],t[k]
        insere(t,k-1)

def tri_eleve2(t) :
    if i < len(t) :
        insere(t,i)
        tri_ins3(t,i+1)
```

```
def tri_eleve3(t):
    n = len(t)
    def insere_eleve3(i) :
        k = i
        while k > 0 and t[k]<t[k-1] :
            t[k],t[k-1]=t[k-1],t[k]
            k = k-1
    for i in range(n) :
        insere_eleve3(i)
```

```
def tri_eleve4(t) :
    n = len(t)
    i = 0
    while i < n :
        k = i
        while k > 0 and t[k]<t[k-1] :
            t[k],t[k-1]=t[k-1],t[k]
            k = k-1
        i = i + 1
```

23. Pour que chacune puisse apprendre des erreurs des autres, rédiger une synthèse succincte à destination des élèves de première NSI dont vous venez de corriger les productions. L'idée est de s'appuyer sur les erreurs observées pour faire des rappels généraux sur les points clés à vérifier dans un programme afin qu'il fonctionne correctement et efficacement.

Éléments de réponse

- **Terminaison :** si vous choisissez d'écrire des fonctions récursives, il est crucial que vous vous assuriez de la terminaison de votre programme en vérifiant en particulier la présence d'un cas de base. De même, la condition de sortie des boucles `while` doit reposer sur un variant de boucle à expliciter. Cela permet d'éviter les oublis d'incrément, de décrément d'indices de conditions de sortie de boucle.
- **Insérer au plus tôt :** l'itération permettant l'insertion ne nécessite pas toujours de parcourir tous les éléments de `k` à `0`. Dans une version itérative, on préfère donc utiliser une boucle `while`.
- **Sorties d'indices :** lors des parcours de listes, vous devez vous assurer que les accès à des éléments de la liste sont tous valides, en particulier aux limites.
- **Fonctions auxiliaires :** si vous choisissez d'utiliser une fonction auxiliaire, attention à la cohérence entre valeur de retour et usage de cette fonction.

Voici une implémentation possible du tri par insertion. La première boucle est un `for` car on connaît la liste des valeurs à parcourir à l'avance. La deuxième boucle est un `while` pour éviter des parcours inutiles comme discuté précédemment. Dans un soucis de modularité, on peut utiliser une fonction auxiliaire pour chaque étape d'insertion. Cette fonction agit par effets de bord.

```
def insere(t,i) :
    k = i
    while k > 0 and t[k]<t[k-1] :
        t[k],t[k-1]=t[k-1],t[k]
        k = k-1

def tri_ins(t):
    n = len(t)
    for i in range(n) :
        insere(t,i)
```

24. Une élève vous demande pourquoi on n'utilise pas un algorithme de dichotomie pour améliorer la complexité de l'insertion des éléments. Que lui répondez-vous ?

Éléments de réponse

La recherche dichotomique de l'indice `ind` où insérer `t[k]` dans la portion `t[:k]` permet effectivement d'obtenir un tri dont le nombre de comparaisons est un $O(n \log n)$ où n est la longueur de la liste à trier. Malheureusement, une fois

cet indice calculé, il faut encore décaler tous les éléments entre les indices `ind` et `k`. La complexité temporelle serait alors tout de même un $O(n^2)$. Cette amélioration serait vraiment intéressante surtout dans le cas où la comparaison entre deux éléments serait très coûteuse.

5.2 Tri par sélection

25. On rappelle succinctement le fonctionnement du tri par sélection au travers du pseudo-code suivant :

```
def tri_sel(t) :
    pour k de 0 à n-1
        sélectionner l'indice i0 du minimum des éléments entre les indices de k à n-1
        échanger t[k] et t[i0]
```

Écrire l'énoncé d'un devoir maison permettant de faire étudier cet algorithme à vos élèves de première NSI. Cet énoncé devra les guider pour :

- découvrir cet algorithme,
- l'implémenter,
- prouver sa correction,
- estimer sa complexité en nombre de comparaisons.

Attention, le contenu de cet énoncé devra se suffire à lui-même, et ne dépendre d'aucune ressource externe ou interventions spécifiques de votre part.

Éléments de réponse

L'objectif de cette activité est d'implémenter le tri par sélection. Le principe est intuitif. On commence par chercher l'indice de la valeur minimale de la liste et on échange la valeur de cette case avec la case d'indice 0. Cet élément est à sa place définitive. Il nous reste à trier le reste des éléments en procédant de la même façon : à chaque étape, on sélectionne le minimum des éléments restants dans la portion `t[k:]` et on l'échange avec `t[k]`.
Voici un exemple d'exécution de ce tri sur le tableau

4	6	2	5	8	1	3	7

:

4	6	2	5	8	**1**	3	7	l'indice du minimum de `t[0:]` vaut 5
1	6	**2**	5	8	4	3	7	l'indice du minimum de `t[1:]` vaut 2
1	2	6	5	8	4	**3**	7	l'indice du minimum de `t[2:]` vaut 6
1	2	3	5	8	**4**	6	7	l'indice du minimum de `t[3:]` vaut 5
1	2	3	4	8	**5**	6	7	l'indice du minimum de `t[4:]` vaut 5
1	2	3	4	5	8	**6**	7	l'indice du minimum de `t[5:]` vaut 6
1	2	3	4	5	6	8	**7**	l'indice du minimum de `t[6:]` vaut 7
1	2	3	4	5	6	7	8	

(a) Dérouler l'algorithme étape par étape de la même façon pour la liste suivante :

5	3	8	1	6	2	7	4

(b) Implémentation de l'algorithme

i. Écrire une fonction `selection` telle que l'appel `selection(t,k)` renvoie l'indice de l'élément minimal parmi les éléments de l'indice `k` à la fin de la liste `t` en utilisant une boucle `for` d'indice `i` croissant.

ii. Écrire une fonction `tri_sel` telle que l'appel `tri_sel(t)` trie la liste `t`, en utilisant une boucle `for` pour sélectionner le minimum des éléments restants à chaque étape et l'échanger avec l'élément d'indice `k`. La fonction `tri_sel` ne renvoie aucune valeur, elle agit par effets de bord.

(c) Analyse de l'implémentation

i. Justifier que la fonction `tri_sel` termine.

ii. Pour démontrer la correction de la fonction `selection`, considérons l'invariant suivant :

"à la fin de l'itération d'indice `i`, l'indice stocké est compris entre `k` et `i` et c'est l'indice d'une valeur inférieure ou égale à chacune des valeurs d'indices entre `k` et `i`"

Prouver cet invariant et l'utiliser pour conclure quant à la correction de la fonction `selection`.

iii. Proposer un invariant de boucle suffisant pour démontrer la correction de la fonction `tri_sel`. En supposant cet invariant prouvé, conclure quant à la correction de la fonction `tri_sel`.

(d) Complexité

i. Quelle est la complexité de la fonction `selection` ?

ii. Quelle est la complexité de la fonction `tri_sel` ?

iii. Y-a-t-il des cas où la complexité est meilleure comme les listes déjà triées pour le tri par insertion ?

26. À la fin de la séance, une élève vous demande pourquoi on n'utilise pas un algorithme de dichotomie pour la sélection des éléments. Que lui répondez-vous ?

Éléments de réponse

C'est bien de penser "dichotomie" dès que l'on cherche un élément, mais attention à ne pas tout mélanger. La portion dans laquelle on cherche n'est pas triée, ce qui est rédhibitoire pour la recherche dichotomique.

6 Tris par fusion

Cette section s'articule autour du tri partition-fusion du programme de terminale NSI et d'une version simplifiée du tri *Timsort* qui agit également par fusions successives.

6.1 Tri partition-fusion

Lors d'un devoir sur table en classe de terminale NSI, vous avez demandé à vos élèves d'implémenter le tri partition-fusion sans plus de précision. Trois productions d'élèves sont reproduites ci-dessous.

Élève 1 :

```
def fus1(t1,t2):
    n1,n2 = len(t1),len(t2)
    if n1 = 0 : return t2
    if n2 = 0 : return t1
    if t1[0]<t2[0] :
        return [t1[0]]+fus1(t1[1:],t2)
    else :
        return [t2[0]]+fus1(t1,t2[1:])
```

```
def tri_f1(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        return fus1(tri_f1(t[:m]),tri_f1(t[m:]))
```

Élève 2 :

```
def fus2(t1,t2):
    n1,n2 = len(t1),len(t2)
    while i1 < n1 and i2 < n2 :
        if t1[i1]<t2[i2] :
            res.append(t1[i1])
            i1 = i1 + 1
        else :
            res.append(t2[i2])
            i2 = i2 + 1
    if i1 == n1 : res.extend(t2[i2:])
    else : res.extend(t1[i1:])
    return res
```

```
def tri_f2(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        t1,t2 = t[:m],t[m:]
        tri_f2(t1)
        tri_f2(t2)
        return fus2(t1,t2)
```

Élève 3 :

```
def fus3(t,g,m,d):
    aux = []
    i1,i2 = g,m
    while i1<m and i2<d :
        if t[i1]<t[i2] :
            aux.append(t[i1])
            i1 = i1 + 1
        else :
            aux.append(t[i2])
            i2 = i2 + 1
    if i1<m : aux.extend(t[i1:m])
    for i in range(len(aux)):
        t[i+g] = aux[i]
```

```
def tri_f3(t):
    def aux(g,d):
        if g < d-1 :
            m = (g+d)//2
            aux(g,m)
            aux(m,d)
            fus3(t,g,m,d)
    aux(0,len(t))
```

27. Élaborer un barème détaillé pour cette question.

Éléments de réponse

Le barème suivant est proposé avec un point sur chacun des éléments évalués :

- Évaluation de la fusion :
  - syntaxe, initialisations, incrémentations
  - terminaison, cas de base, sortie de boucle
  - correction
  - complexité
- Évaluation du tri :
  - syntaxe, initialisations, incrémentations
  - terminaison, cas de base, sortie de boucle
  - cohérence avec fusion
  - correction
  - complexité

28. Corriger ces trois productions, c'est-à-dire donner la note détaillée selon le barème et l'appréciation associée pour chacune des productions d'élèves. **Les corrections seront données directement sur les productions d'élèves fournies dans le document réponse 1 (Annexe E). Ce document réponse est à rendre avec votre copie.**

Éléments de réponse

Élève 1 : le programme fonctionne, la fusion a une complexité en nombre de comparaisons satisfaisantes, mais avec une mauvaise complexité temporelle ($O(n^2)$ où n est la taille de `t1` et `t2`) à cause des concaténations et des slicings.
évaluation de la fusion : (syntaxe $0,5$; terminaison ok ; correction ok ; complexité $0,5$)
évaluation du tri : (syntaxe ok ; terminaison ok ; cohérence avec fusion ok ; correction ok ; complexité ok)

Élève 2 : `fus2` fonctionne bien, mais `tri_f2` mélange fonction triant une liste "en place" et fonction renvoyant une liste triée contenant les mêmes éléments.
Attention aux initialisations.
évaluation de la fusion : (syntaxe 0,5 ; terminaison ok ; correction ok ; complexité ok)
évaluation du tri : (syntaxe ok ; terminaison ok ; cohérence avec fusion ok ; correction $0,5$; complexité ok)

Élève 3 : le programme fonctionne très bien
évaluation de la fusion : (syntaxe ok ; terminaison ok ; correction ok ; complexité ok)
évaluation du tri : (syntaxe ok ; terminaison ok ; cohérence avec fusion ok ; correction ok ; complexité ok)

6.2 Tri *Timsort* simplifié

On présente ci-dessous une implémentation d'un tri *Timsort* simplifié.

```
def f1(l,a):
    i = a
    while i<len(l)-1 and l[i]<=l[i+1] :
        i += 1
    return i+1

def f2(l):
    i = 0
    res = [0]
    while i<len(l) :
        i = f1(l,i)
        res.append(i)
    return res

def f3(t, debut, milieu, fin) :
    i = debut
    j = milieu
    lt = []
    while i < milieu and j < fin :
        if t[i] < t[j] :
            lt.append(t[i])
            i += 1
        else :
            lt.append(t[j])
            j += 1
    for k in range (i, milieu) : lt.append(t[k])
    for k in range (debut, j) : t[k] = lt[k-debut]

def f4(t,rc):
    nrc = []
    i = 0
    n = len(rc)
    while i < n :
        nrc.append(rc[i])
        if i+2 < n :
            f3 (t, rc[i], rc[i+1], rc[i+2])
            i += 2
        else : i += 1
    return nrc

def f5(t):
    rc = f2(t)
    while len(rc) > 2 :
        rc = f4 (t, rc)
```

29. (a) Proposer des noms explicites pour ces fonctions.
    (b) Présenter le fonctionnement général de ces fonctions tel que vous le feriez à vos élèves.

Éléments de réponse

Réponses aux questions (a) et (b) - Voici une proposition de noms et une explication du fonctionnement de ces fonctions :

`f1` : `detectPortionMax`, elle prend en entrée une liste `t` et un indice `a` et renvoie le plus grand indice `b` tel que `t[a:b]` soit triée.

`f2` : `detectPortionsMax`, elle prend en entrée une liste `t` et renvoie la plus petite liste `res` d'indices telle que `res[0] = 0`, `res[-1] = n` et chaque portion entre les indices successifs est triée.

`f3` : `fusionne`, prend en entrée une liste et 3 indices dans l'ordre croissant tels que les 2 portions entre ces trois indices sont triées et fusionne ces portions pour que l'union des deux soit triée.

`f4` : `etage`, fusionne les portions triées deux par deux et renvoie la nouvelle liste des indices les séparant.

`f5` : `tri`, applique la fonction `etage` jusqu'à ce qu'il n'y ait plus qu'une seule portion, c'est-à-dire `rc = [0,n]`.

(c) Illustrer le principe de ce tri sur un ou plusieurs exemples pour que les élèves puissent l'implémenter par elles-mêmes.

Éléments de réponse

Si `t = [5, 36, 47, 13, 37, 48, 2, 13, 44, 48, 22, 7, 23, 44, 30, 3, 10, 26, 29, 46]`

- à la première étape :

$$\texttt{rc} = [0, 3, 6, 10, 11, 14, 15, 20]$$

- après la première application de `etage` :

$$\texttt{t} = [5, 13, 36, 37, 47, 48, 2, 13, 22, 44, 48, 7, 23, 30, 44, 3, 10, 26, 29, 46]$$
$$\texttt{rc} = [0, 6, 11, 15, 20]$$

- après la seconde application de `etage` :

$$\texttt{t} = [2, 5, 13, 13, 22, 36, 37, 44, 47, 48, 48, 3, 7, 10, 23, 26, 29, 30, 44, 46]$$
$$\texttt{rc} = [0, 11, 20]$$

- et enfin :

$$\texttt{t} = [2, 3, 5, 7, 10, 13, 13, 22, 23, 26, 29, 30, 36, 37, 44, 44, 46, 47, 48, 48]$$
$$\texttt{rc} = [0, 20]$$

30. Quelle est la complexité, dans le pire cas, en nombre de comparaisons de cet algorithme de tri ?

Éléments de réponse

Au pire, `rc` est de longueur n au départ. Chaque étape est de complexité $O(n)$ et la fonction `etage` est appelée $O(\log_2(n))$ fois. La complexité globale est donc un $O(n\log(n))$.

7 Tris sans comparaison

Votre classe de terminale NSI a bien compris les différents algorithmes du tri du programme. Vous décidez donc d'aller plus loin en leur faisant découvrir d'autres algorithmes de tri qui fonctionnent sans comparaison sur des listes d'entiers positifs.

7.1 Tri par dénombrement

Au lieu de comparer les éléments entre eux, le tri par dénombrement compte le nombre d'occurrences de chaque élément. Une fois les nombres d'occurrences obtenus, la liste est reconstituée dans l'ordre croissant en y plaçant tous les éléments présents dans l'ordre croissant avec le bon nombre d'occurrences.

Par exemple, pour trier $[1, 7, 5, 3, 7, 7, 1, 3, 3, 7, 5]$, on compte qu'il y a :

— 2 occurrences de 1

— 3 occurrences de 3

— 2 occurrences de 5

— 4 occurrences de 7

On peut alors reconstituer la liste triée : $[1, 1, 3, 3, 3, 5, 5, 7, 7, 7, 7]$.

Vous avez posé la question suivante à vos élèves :

Écrire la fonction Python `tri_den` *qui prend en paramètre une liste* `t` *d'entiers positifs, ne renvoie aucune valeur mais y applique le tri par dénombrement. La fonction proposée aura une complexité temporelle en $O(n + vmax)$ où n est le nombre d'éléments dans* `t` *et vmax est la valeur maximale dans* `t`.

Une élève vous rend le travail suivant :

```
def compte_occu(t,elt):
    c = 0
    for x in t :
        if x == elt : c = c+1
    return c

def tri_den_el(t):
    occ = [0]*10000
    res = []
    for elt in range(10000):
        occ[elt] = compte_occu(t,elt)
        res = res + [elt]*occ[elt]
    return res
```

31. Lister les indications à lui donner pour qu'elle puisse répondre correctement à votre question.

Éléments de réponse

La valeur maximale présente dans ta liste n'est pas nécessairement inférieure à 10 000. Le programme attendu doit pouvoir déterminer lui-même cette valeur maximale.
L'instruction `res = res + [elt]*occ[elt]` est coûteuse parce qu'elle provoque la recopie de toute la liste `res`. Enfin, la consigne t'impose de modifier la liste `t` pour qu'elle soit triée et non de construire une nouvelle liste.
De plus, pour respecter la complexité demandée, il n'est pas possible de faire un parcours pour chaque valeur possible présente dans la liste. L'idée de construire un tableau `occ` est bonne, mais on pourrait directement écrire une fonction `compte_occus(t)` qui en un seul parcours de `t` renvoie le tableau `occ` des nombres d'occurrences de tous les éléments.

32. Écrire, sans commenter, le programme que vous espérez qu'elle écrive.

Éléments de réponse

La complexité attendue sous-entend l'usage d'une liste.

```
def compte_occus(t):
    vmax = 0
    for x in t :
        if vmax < x : vmax = x
    occ = (vmax+1)*[0]
    for x in t :
        occ[x] += 1
    return occ

def tri_den(t):
    occ = compte_occus(t)
    i = 0
    j = 0
    while i<len(t) :
        if occ[j]>0 :
            t[i] = j
            occ[j] -= 1
            i += 1
        else :
            j += 1
```

33. La complexité temporelle de cet algorithme contredit-elle la borne de complexité admise en introduction de ce sujet ? Pourquoi ?

Éléments de réponse

Pas d'incohérence car la borne inférieure a été prouvée pour les algorithmes de tri par comparaisons. Notons que la restriction aux entiers positifs n'est pas significative car la preuve de la borne inférieure reste valable dans ce cadre.

34. Cet algorithme de tri est-il toujours plus efficace, en termes de complexité temporelle, que le tri fusion dont la complexité temporelle est en $O(n \log n)$? Justifier.

Éléments de réponse

Cet algorithme de tri n'est pas toujours plus efficace que le tri fusion. En particulier, si $vmax = n^2$, alors pour un n suffisamment grand, il vaut mieux utiliser le tri fusion.

7.2 Tri par baquets

Dans cette partie, le tri par baquets est étudié, le principe est le suivant :

- on travaille avec des baquets numérotés de 0 à 9 (ces baquets sont implémentés par une liste de 10 listes) ;
- pour débuter, chaque élément de la liste initiale est placé dans le baquet numéroté par son chiffre des unités ;
- à chaque étape, les éléments sont parcourus baquet par baquet en partant de l'indice 0 du baquet 0 et rangés à nouveau dans les baquets en utilisant le chiffre suivant ;
- l'algorithme est terminé lorsque tous les éléments sont dans le baquet 0.

Par exemple : si

$$\texttt{t} = [8295, 7971, 8806, 1203, 2916, 6231, 1112, 6131, 538, 3832, 1813, 6863, 6200, 9449, 1905, 1749]$$

- au départ, les éléments sont rangés dans les baquets par chiffres des unités :

$$[[6200], [7971, 6231, 6131], [1112, 3832], [1203, 1813, 6863], [], [8295, 1905], [8806, 2916], [], [538], [9449, 1749]]$$

- puis on les place par chiffre des dizaines en préservant l'ordre sur les unités dans chaque baquet grâce à l'ordre de parcours des éléments :

$$[[6200, 1203, 1905, 8806], [1112, 1813, 2916], [], [6231, 6131, 3832, 538], [9449, 1749], [], [6863], [7971], [], [8295]]$$

- puis par chiffre des centaines en respectant l'ordre précédent dans chaque baquet :

$$[[], [1112, 6131], [6200, 1203, 6231, 8295], [], [9449], [538], [], [1749], [8806, 1813, 3832, 6863], [1905, 2916, 7971]]$$

- puis par chiffre des milliers :

$$[[538], [1112, 1203, 1749, 1813, 1905], [2916], [3832], [], [], [6131, 6200, 6231, 6863], [7971], [8295, 8806], [9449]]$$

- à la dernière étape, tous les éléments se retrouvent dans le baquet 0 :

$$[[538, 1112, 1203, 1749, 1813, 1905, 2916, 3832, 6131, 6200, 6231, 6863, 7971, 8295, 8806, 9449], [], [], [], [], [], [], [], [], []]$$

35. Écrire une fonction Python `ch` qui prend en paramètre deux entiers et telle que l'appel `ch(k,i)` renvoie le `i`-ème chiffre de l'entier `k`. Par exemple, `ch(1345,2) = 4`, `ch(1345,4) = 1` et `ch(1345,5) = 0`. Cette fonction pourra être utilisée par vos élèves pour l'implémentation du tri par baquets.

Éléments de réponse

Plusieurs possibilités plus ou moins élégantes pour cette question :

```
def ch(k,i):
    return (k % 10**(i))//10**(i-1)
```

```
def ch(k,i):
    l = str(k)
    if i <= len(l) : return int(l[-i])
    else : return 0
```

36. Vous avez fourni une implémentation à trous de l'algorithme de tri par baquets à vos élèves. En particulier, la fonction Python `tri_baquets` prend en paramètre une liste `t`, applique le tri par baquets à `t` et renvoie une liste triée contenant les mêmes éléments que `t`.

```
def etape(bacs1,bacs2,i):
    for b in bacs1 :
        for x in b :
            bacs2[...........].append(...........)

def vider(bacs):
    for i in range(10):
        bacs[i] = ...........

def tri_baquets(t) :
    n = len(t)
    bacs1 = [[] for i in range(10)]
    bacs2 = [[] for i in range(10)]
    for x in t:
        bacs1[...........].append(...........)
    i = 2
    while len(bacs1[0]) != n :
        etape(bacs1,bacs2,i)
        bacs1,bacs2 = ...........
        vider(...........)
        i += 1
    return ...........
```

Compléter ce que vous attendez que les élèves écrivent dans les trous sans commenter. **Vous compléterez cet algorithme à trous fourni dans le document réponse 2 (Annexe F). Ce document réponse est à rendre avec votre copie.**

Éléments de réponse

On choisit d'utiliser 2 séries de baquets, mais il est évidemment possible de reconstituer la liste à chaque étape et de réutiliser les mêmes baquets.

```
def etape(bacs1,bacs2,i):
    for b in bacs1 :
        for x in b :
            bacs2[ch(x,i)].append(x)

def vider(bacs):
    for i in range(10):
        bacs[i] = []

def tri_baquets(t) :
    n = len(t)
    bacs1 = [[] for i in range(10)]
    bacs2 = [[] for i in range(10)]
    for x in t:
        bacs1[ch(x,1)].append(x)
    i = 2
    while len(bacs1[0]) != n :
        etape(bacs1,bacs2,i)
        bacs1,bacs2 = bacs2,bacs1
        vider(bacs2)
        i += 1
    return bacs1[0]
```

37. Quelle est la complexité temporelle de ce tri ?

Éléments de réponse

La fonction `etape` est de complexité $O(n)$ et est appelée $O(\log_{10}(vmax))$ fois. La complexité globale est en $O(\log(vmax)*n)$.

38. Dans quels cas utiliser cet algorithme plutôt que le tri par dénombrement ?

Éléments de réponse

Si $vmax$ n'est pas un $O(n\log(n))$ et que n est suffisamment grand, il est préférable d'utiliser ce tri plutôt que le tri par dénombrement.

39. Cet algorithme de tri est-il toujours plus efficace, en termes de complexité temporelle, que le tri fusion dont la complexité temporelle est en $O(n\log n)$?

Éléments de réponse

Si $vmax$ n'est pas un $O(n)$ et que n est suffisamment grand, cet algorithme de tri est moins efficace que le tri fusion.

A Extrait du fichier Donnees_capteurs_22_09_21.csv (ouvert dans un tableur)

Le séparateur du fichier .csv est " ;".

	A	B	C	D	E	F
1	Identifiant_capteur	Piece	Num_mesure	Ouverture	Presence	Temperature
2	1	Salon	1	False	False	18,7
3	3	Cuisine	1	False	False	18,8
4	2	Chambre 1	1	True	False	17,5
5	5	Salle de bair	1	False	False	19,1
6	6	Chambre 2	1	False	True	16,6
7	4	Entree	1	False	False	17,4
8	1	Salon	2	False	False	18,7
9	3	Cuisine	2	False	False	18,8
10	2	Chambre 1	2	True	False	17,5

B Description de fonctions utiles de la bibliothèque pandas

- pandas.read_csv(nom,sep=...) prend en argument une chaîne contenant l'adresse du fichier csv à lire et un chaîne contenant le séparateur du fichier csv, et retourne une instance de type dataframe, initialisée avec l'ensemble des données présentes dans le fichier. Lors de la création du dataframe, un index est associé à chacune des lignes du fichier (en commençant à partir de 0).

Par la suite, pour illustrer l'utilisation de certaines fonctions, on suppose que la variable "df" pointe vers le dataframe obtenu à partir du fichier Donnees_capteurs_22_09_21.csv.

- df.loc(index_ligne,index_colonne) retourne, sous forme d'un objet de type dataframe, certaines lignes et colonnes du dataframe df. L'index des colonnes est le nom des colonnes dans le fichier csv, et l'index des lignes est l'index créé lors de l'initialisation du dataframe à la lecture du fichier.
  Par exemple, print(df.loc[0,'Temperature']) affichera la température de la 1ère ligne du fichier.
  Il est également possible d'effectuer des tests sur le contenu d'un dataframe.
  Par exemple, print(df['Piece']=='Salon') affichera, sous forme de booléens, pour chaque ligne du fichier, si la pièce mentionnée est un salon.
  Ces deux techniques peuvent être conjointement utilisées pour filtrer des lignes à partir de tests.
  Par exemple, print(df.loc[df['Piece']=='Salon', :]) affichera toutes les lignes qui concernent le salon.
  Il est possible de combiner plusieurs facteurs de filtrage en utilisant un "et"(&) ou un "ou"(|).
  Par exemple, print(df.loc[(df['Piece']=='Salon')|(df['Piece']=='Cuisine'),'Presence']) affichera la colonne présence pour les lignes correspondant au salon ou à la cuisine.

- Il est possible de retourner le maximum (méthode max), le minimum (méthode min) et la moyenne (méthode mean) d'une donnée d'une dataframe.
  Par exemple, print(df.loc[:,'Temperature'].mean()) affichera la température moyenne mesurée sur l'ensemble des lignes du fichier.

- Pour retourner le contenu trié d'un dataframe, on utilise la méthode dataframe.sort_values(column,ascending=...). column est une chaîne indiquant la colonne utilisée pour le tri. Si ascending vaut True (valeur par défaut), le tri est croissant, s'il vaut False, il est décroissant.
  Par exemple, df.sort_values(by=["Piece"], ascending=False) affiche les lignes du fichier triées par Piece dans l'ordre alphabétique inversé (en commençant par Z).

C Extraits des fichiers pour le traitement des données sous forme de table

C.1 Extrait du fichier positifs.csv

```
num_SS;age;jour_test;mois_test;sexe;vaccine
1630580265006;58;2;7;H;True
2750846580156;46;2;7;F;False
1950412410023;26;3;7;H;False
2570235820132;64;3;7;F;False
```

C.2 Extrait du fichier restaurants.csv

```
id_enregistrement;id_restaurant;jour_repas;mois_repas;tranche_horaire;num_table;num_SS
1;324;30;6;2;5;1850659240032
2;324;30;6;2;5;2860372541115
3;796;1;7;1;8;1950412410023
4;796;1;7;1;8;1951012410056
5;796;1;7;1;8;1951232564031
```

D Extrait de la base de données

Films			
Id	***Titre***	***Annee***	***Pays***
12546	Entre les murs	2008	France
2506	Fight Club	1999	USA
67895	La cité de la peur	1994	France
54781	Yes Day	2021	USA

Clients				
Id	***Nom***	***Prenom***	***AdresseMail***	***TypeAbonnement***
1761	Dupont	Martin	dm@capes.fr	Basic
564	Tartanpion	Lucie	TLucie@mail.com	Premium
153	Durant	Hervé	RVDurant@capes.fr	Premium
789	Martin	Nina	Nina@pro.fr	Basic

Visionnage					
Id	***IdClient***	***IdFilm***	***Date***	***Heure***	***TempsVisionnage***
123456	1761	5468	11/08/2021	10 :45	35
93720	2456	14803	09/11/2020	21 :10	124
115986	1602	14972	27/03/2021	08 :30	2
78439	2395	6730	16/05/2020	22 :35	48

Facturation				
Id	***IdClient***	***Montant***	***Date***	***Statut***
2761	1254	7.90	12/07/2021	Payee
1065	1542	7.90	05/08/2020	Payee
1753	864	11.90	03/04/2021	En attente
1334	741	11.90	09/10/2020	Payee

E Document Réponse 1 : Correction de trois productions d'élèves

Document réponse de la question 28 de la section 6.1 (Problème 2). **Ce document réponse doit être rendu avec votre copie.**

Élève 1 :

```
def fus1(t1,t2):
    n1,n2 = len(t1),len(t2)
    if n1 = 0 : return t2
    if n2 = 0 : return t1
    if t1[0]<t2[0] :
        return [t1[0]]+fus1(t1[1:],t2)
    else :
        return [t2[0]]+fus1(t1,t2[1:])
```

```
def tri_f1(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        return fus1(tri_f1(t[:m]),tri_f1(t[m:]))
```

Élève 2 :

```
def fus2(t1,t2):
    n1,n2 = len(t1),len(t2)
    while i1 < n1 and i2 < n2 :
        if t1[i1]<t2[i2] :
            res.append(t1[i1])
            i1 = i1 + 1
        else :
            res.append(t2[i2])
            i2 = i2 + 1
    if i1 == n1 : res.extend(t2[i2:])
    else : res.extend(t1[i1:])
    return res
```

```
def tri_f2(t):
    if len(t) <2 :
        return t
    else :
        m = len(t)//2
        t1,t2 = t[:m],t[m:]
        tri_f2(t1)
        tri_f2(t2)
        return fus2(t1,t2)
```

Élève 3 :

```
def fus3(t,g,m,d):
    aux = []
    i1,i2 = g,m
    while i1<m and i2<d :
        if t[i1]<t[i2] :
            aux.append(t[i1])
            i1 = i1 + 1
        else :
            aux.append(t[i2])
            i2 = i2 + 1
    if i1<m : aux.extend(t[i1:m])
    for i in range(len(aux)):
        t[i+g] = aux[i]
```

```
def tri_f3(t):
    def aux(g,d):
        if g < d-1 :
            m = (g+d)//2
            aux(g,m)
            aux(m,d)
            fus3(t,g,m,d)
    aux(0,len(t))
```

F Document Réponse 2 : Algorithme de tri par baquets à trous à compléter

Document réponse de la question 36 de la section 7.2. **Ce document réponse doit être rendu avec votre copie.**

```
def etape(bacs1,bacs2,i):
    for b in bacs1 :
        for x in b :
            bacs2[.....].append(.....)

def vider(bacs):
    for i in range(10):
        bacs[i] = .....

def tri_baquets(t) :
    n = len(t)
    bacs1 = [[] for i in range(10)]
    bacs2 = [[] for i in range(10)]
    for x in t:
        bacs1[.....].append(.....)
    i = 2
    while len(bacs1[0]) != n :
        etape(bacs1,bacs2,i)
        bacs1,bacs2 = .....
        vider(.....)
        i += 1
    return .....
```