COMPOSITION D'ITC n°3

(Durée : 2 heures)

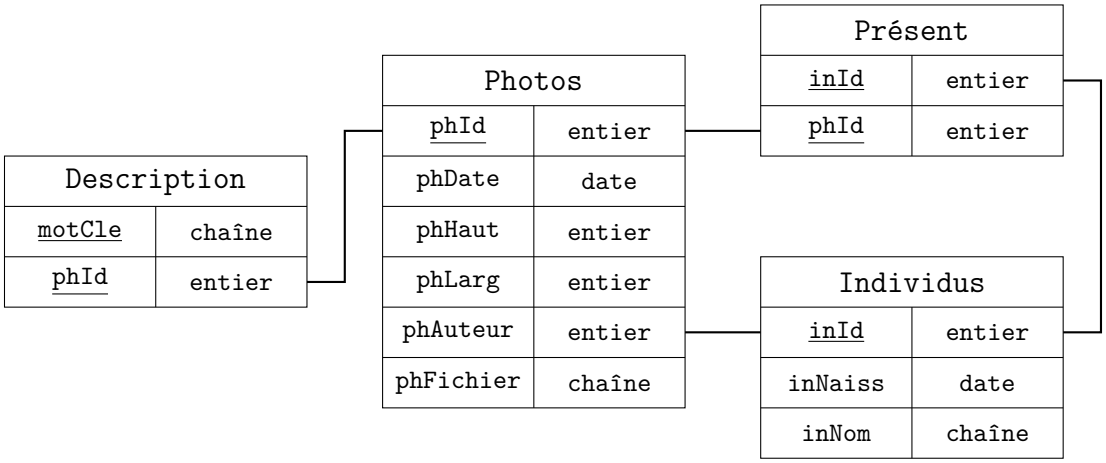
L'utilisation de la calculatrice **est autorisée** pour cette épreuve.

⋆ ⋆ ⋆

Le sujet est composé de deux parties indépendantes.

1 Base de données photographique

On dispose d'une base de données répertoriant des photos avec diverses informations. La base données est représentée par le schéma ci-dessous :



La clé primaire de chaque table est signalée par un soulignement des attributs concernés.

On a pour chaque table représentant une entité la description de ses attributs :

- `Photos` :
  - `phId` : identifiant de la photo
  - `phDate` : date et heure de la photo (on supposera le type `date` compatible avec les opérations de comparaison, le minimum, le maximum)
  - `phHaut`, `phLarg` : dimensions de la photo en pixels
  - `phAuteur` : identifiant de la personne ayant pris la photo
  - `phFichier` : nom du fichier contenant la photo
- `Individus` :
  - `inId` : identifiant de l'individu
  - `inNaiss` : date de naissance de l'individu
  - `inNom` : nom de l'individu

Les tables `Présent` et `Description` représentent des relations entre les entités. `Présent` indique les personnes apparaissant sur les photos, et `Description` indique les mots-clé associés aux photos.

Question 1 En justifiant à l'aide des clés primaires, déterminer

1. si une personne peut apparaitre sur plusieurs photos,
2. si plusieurs personnes peuvent apparaitre sur la même photo.

Question 2 Écrire une requête donnant les identifiants des photos dont le format est **exactement** 16/9.

Question 3 Écrire une requête donnant les noms des individus apparaissant dans au moins un *selfie*, c'est-à-dire une photos sur laquelle l'auteur de la photo apparait.

Question 4 Écrire une requête donnant les identifiants des photos sur lesquelles personne n'apparait.

Question 5 Écrire une requête donnant les noms des fichiers contenant des photos décrites par au moins 5 mots-clé.

2 Clustering en dimension 1

On souhaite mettre en place du soutien différencié au travail dans une classe de lycée. Pour ce faire, on souhaite séparer les élèves en plusieurs groupes de niveaux homogènes pour leur proposer des exercices adaptés à leur niveau. On dispose pour cela des notes des élèves et on veut les regrouper selon la similitude de leurs notes.

Formellement, on dispose d'un ensemble E de N réels $x_0 \leqslant x_1 \leqslant \ldots \leqslant x_{N-1}$. On cherche à déterminer une partition de $[\![0, N-1]\!]$ en $K \leqslant N$ sous-ensembles $\mathcal{P} = \{C_0, C_1, \ldots, C_{K-1}\}$ non vides tels que le score

$$S(\mathcal{P}) = \sum_{i=0}^{K-1} \sum_{j \in C_i} (x_j - \mu_i)^2$$

soit minimal, où $\mu_i = \dfrac{1}{|C_i|} \sum\limits_{j \in C_i} x_j$ est la moyenne des éléments correspondant à la classe C_i. Autrement dit, on veut minimiser la somme des carrés des écarts de chaque élément à la moyenne de sa classe.

Par exemple, pour $E = \{1, 2, 3, 5, 8, 10, 14, 15, 18\}$, une solution optimale pour $K = 3$ est donnée par la partition suivante :

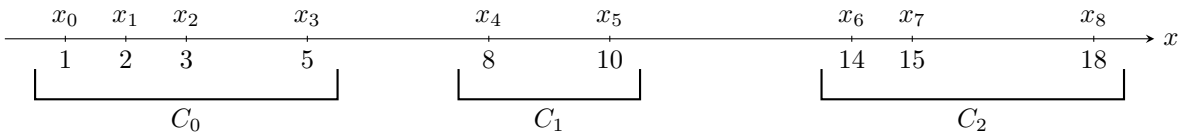


FIGURE 1 – Une partition optimale $\mathcal{P} = \{\{0, 1, 2, 3\}, \{4, 5\}, \{6, 7, 8\}\}$ de $[\![0, 8]\!]$ pour $K = 3$ et l'ensemble $E = \{1, 2, 3, 5, 8, 10, 14, 15, 18\}$, de score environ $19,42$.

Si $\{C_0, C_1, \ldots, C_{K-1}\}$ est une partition solution du problème, on remarque qu'on peut supposer sans perte de généralité que les classes sont rangées par ordre croissant, c'est-à-dire que pour $i < i'$ et $j \in C_i$, $j' \in C_{i'}$, alors $x_j \leqslant x_{j'}$. On supposera cette hypothèse vérifiée pour l'ensemble des partitions du problème.

2.1 Préliminaires

Question 6 Comment trouver une solution au problème lorsque $K = N$? Lorsque $K = N - 1$? Justifier.

Question 7 Appliquer l'algorithme des K-moyennes sur l'ensemble E de l'exemple précédent. On prendra $K = 3$, et on initialisera les barycentres à $b_0 = 1$, $b_1 = 8$ et $b_2 = 10$. On donnera les détails des étapes de calculs jusqu'à convergence de l'algorithme.

On accepte une représentation graphique pour la description des étapes de calcul.

On cherche à trier une liste `L` en Python. Pour ce faire, on propose les fonctions suivantes :

```
def fusion(L1, L2):
    L = []
    i1, i2, n1, n2 = 0, 0, len(L1), len(L2)
    for _ in range(n1 + n2):
        if L1[i1] <= L2[i2]:
            L.append(L1[i1])
            i1 += 1
        else:
            L.append(L2[i2])
    return L

def tri_fusion(L):
    n = len(L)
    if n < 1:
        return L
    L1, L2 = L[0:n // 2], L[n // 2:n]
    return fusion(tri_fusion(L1), tri_fusion(L2))
```

Question 8 Le code précédent comporte trois erreurs. Préciser les lignes où apparaissent ces erreurs et un moyen de les corriger. On ne demande pas de recopier tout le code.

Pour la suite de cette partie, on supposera que la liste `E` représentant un ensemble E sera toujours triée.

Pour $E = \{x_0, \dots, x_{N-1}\}$ et $0 \leqslant i < j \leqslant N$, on note $S_{\text{emc}}(i, j)$ (pour « somme des écarts à la moyenne au carré ») la valeur

$$S_{\text{emc}}(i,j) = \sum_{k=i}^{j-1} (x_k - \mu)^2$$

où μ est la moyenne des éléments de $\{x_i, x_{i+1}, \dots, x_{j-1}\}$.

Question 9 Écrire une fonction `moyenne(E, i, j)` qui prend en argument une liste E de N valeurs $\{x_0, \dots, x_{N-1}\}$ triées et deux indices $0 \leqslant i < j \leqslant N$ et renvoie la moyenne des éléments de $\{x_i, x_{i+1}, \dots, x_{j-1}\}$.

Question 10 Écrire une fonction `somme_emc(E, i, j)` qui prend en argument une liste E de N valeurs triées et deux indices $0 \leqslant i < j \leqslant N$ et calcule et renvoie $S_{\text{emc}}(i, j)$.

On représente une partition $\mathcal{P} = \{C_0, C_1, \dots, C_{K-1}\}$ de $[\![0, N-1]\!]$ par une liste `P` de taille K telle que `P[i]` est le plus petit élément de C_i. Avec l'hypothèse faite précédemment sur les C_i, la liste `P` doit nécessairement être triée. On remarque, de plus, que `P[0]` vaut toujours 0. Par exemple, la partition $\{\{0,1,2,3\},\{4,5\},\{6,7,8\}\}$ donnée dans l'exemple de la figure 1 est représentée par `[0, 4, 6]`.

Question 11 Dans cette question, on suppose $N = 9$. Quelle est la liste `P` associée à la partition $\mathcal{P} = \{\{0,1,2\},\{3,4\},\{5,6,7\},\{8\}\}$? Quelle est la partition associée à la liste `P = [0, 1, 4, 5]` ?

Question 12 Écrire une fonction `score(E, P)` qui prend en argument un ensemble `E` de taille N et une partition `P` de $[\![0, N-1]\!]$ et renvoie le score $S(\mathcal{P})$ tel qu'il a été défini précédemment.

Question 13 En déduire une fonction `clustering3(E)` qui prend en argument un ensemble `E` de taille $N \geqslant 3$ et renvoie une partition `P` de $[\![0, N-1]\!]$ de taille $K = 3$ de score minimal.

Question 14 Quelle est la complexité temporelle de la fonction précédente ? Justifier.

2.2 Clustering hiérarchique ascendant

On cherche dans cette sous-partie à calculer une solution pas nécessairement optimale par une approche gloutonne dite « hiérarchique ascendante » ou CHA. Pour partitionner un ensemble $E = \{x_0, \ldots, x_{N-1}\}$ en K classes, l'idée est la suivante :

- on crée N classes, chacune correspondant à un singleton d'un élément de E ;
- tant qu'il reste plus que K classes, on fusionne les deux classes les plus proches, c'est-à-dire celles qui ont leurs moyennes les plus proches. En cas d'égalité, on fusionne celles qui ont les moyennes les plus basses.

Question 15 Écrire une fonction `classes_plus_proches(E, P)` qui prend en argument un ensemble `E` de taille N et une partition `P` de $[\![0, N-1]\!]$ de taille K et renvoie un indice i_{opt} tel que $C_{i_{\text{opt}}}$ et $C_{i_{\text{opt}}+1}$ sont les classes les plus proches au sens défini ci-dessus.

Question 16 Écrire une fonction `fusion_classes(P, iopt)` qui prend en argument une partition `P` de $[\![0, N-1]\!]$ de taille K et un indice $0 \leqslant i_{\text{opt}} < K-1$ et renvoie une partition de $[\![0, N-1]\!]$ de taille $K-1$ où les classes d'indices i_{opt} et $i_{\text{opt}}+1$ ont été fusionnées.

Question 17 En déduire une fonction `CHA(E, K)` qui calcule et renvoie une partition de taille K d'un ensemble E selon l'algorithme de clustering hiérarchique ascendant.

2.3 Solution optimale en programmation dynamique

Pour $n \in [\![0, N]\!]$ et $k \in [\![1, K]\!]$, on note $D(n, k)$ le score minimal possible d'une partition de $\{x_0, \ldots, x_{n-1}\}$ en k classes non vides.

Question 18 Que vaut $D(n, k)$ lorsque $k = 1$?

Question 19 Montrer que pour $n > 0$ et $k > 1$, $D(n, k) = \min_{i=k-1}^{n-1} (D(i, k-1) + S_{\text{emc}}(i, n))$.

Question 20 En déduire une fonction `clustering_dynamique(E, K)` qui calcule le score minimal possible d'une partition de E en K classes non vides.

Question 21 Déterminer la complexité temporelle de la fonction précédente.

Question 22 Expliquer en français comment modifier la fonction précédente pour qu'elle renvoie une partition optimale plutôt que le score minimal. On ne demande pas de coder cette solution.

On cherche à améliorer la complexité temporelle de la solution précédente en effectuant des précalculs. Pour $0 \leqslant i < n \leqslant N$, notons $\mu(i, n)$ la moyenne des éléments de $\{x_i, \ldots, x_{n-1}\}$. On admet la formule suivante :

$$S_{\text{emc}}(0, n+1) = S_{\text{emc}}(0, n) + \frac{n}{n+1}(x_n - \mu(0, n))^2$$

Question 23 Décrire en français ou pseudo-code un moyen de calculer l'ensemble des $S_{\text{emc}}(i, n)$ pour $0 \leqslant i < n \leqslant N$ en complexité $\mathcal{O}(N^2)$. On détaillera les formules de récurrence utilisées.

Question 24 En déduire une fonction `calcul_Semc(E)` qui prend en argument l'ensemble E et renvoie un dictionnaire `Semc` tel que pour $0 \leqslant i < n \leqslant N$, `Semc[(i, n)]` est égal à $S_{\text{emc}}(i, n)$.

⋆ ⋆ ⋆