TP/TD D'INFORMATIQUE
Révisions

Exercice 1 : Matrices, SQL (CCP Maths 2023)

Dans cet exercice d'informatique commune, on se propose d'écrire des algorithmes dans le but de faire du calcul matriciel, en particulier sur des matrices d'adjacence de graphe.

Notation

Les matrices sont carrées et représentées par des listes dont les éléments correspondent aux lignes de la matrice. Par exemple, la matrice $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ est représentée par la liste `[[1, 2], [3, 4]]`.

Dans la suite, pour définir la matrice d'adjacence $A \in \mathcal{M}_n(\mathbb{R})$ d'un graphe ayant n sommets, on numérote ses sommets de 0 à $n-1$.

1. Écrire une fonction `produit(A,B)` prenant en arguments deux matrices carrées A et B de mêmes dimensions et qui renvoie AB le produit de la matrice A et de la matrice B.
2. Écrire une fonction `oriente(A)` prenant en argument la matrice d'adjacence A d'un graphe et qui retourne `True` si le graphe est orienté et `False` sinon.
3. On admet que le nombre de chemins de longueur p reliant i à j dans un graphe de matrice d'adjacence A est égal au coefficient d'indice (i, j) de la matrice A^p.
   Écrire une fonction `distance(A,i,j)` où A est la matrice d'adjacence d'un graphe et qui renvoie le nombre minimal d'arêtes que l'on doit parcourir pour atteindre le sommet j depuis le sommet i (on suppose qu'un tel chemin existe).

On considère deux tables CLIENTS et PARTENAIRES. La première contient des informations sur les clients et la deuxième permet d'identifier qui sont les partenaires des clients.
La table CLIENTS contient les attributs suivants :

- `id` : identifiant d'un individu (entier), clé primaire ;
- `nom` (chaîne de caractères) ;
- `prenom` (chaîne de caractères) ;
- `ville` (chaîne de caractères) ;
- `email` (chaîne de caractères).

La table PARTENAIRES contient les attributs suivants :

- `id` : identifiant de suivi (entier), clé primaire ;
- `id_client` : identifiant du client représenté par l'attribut `id` dans la table CLIENTS ;
- `partenaire` : nom du partenaire (chaîne de caractères).

4. Écrire une requête SQL permettant d'extraire les identifiants de tous les clients provenant de la ville de ≪Toulouse≫.
5. Écrire une requête SQL permettant d'extraire les emails de tous les clients ayant ≪SCEI≫ comme partenaire.
6. Écrire une requête SQL affichant pour chaque ville le nombre de clients présents dans cette ville.

Exercice 2 : Représentation de nombres

1. Donner la représentation de 38 en binaire sur 8 bits.
2. Écrire une fonction `binaire` prenant en entrée k et n et renvoyant la liste correspondant à l'écriture de k sur n bits.
3. Donner la représentation de -38 en complément à deux sur 8 bits.
4. Écrire 43 en hexadécimal (base 16).

Exercice 3 : Fonctions usuelles

1. Écrire une fonction testant si un élément appartient à une liste. Donner sa complexité dans le meilleur et le pire cas.
2. Écrire une fonction renvoyant le minimum des éléments d'une liste. Donner sa complexité dans le meilleur et le pire cas.
3. Écrire une fonction testant si un élément appartient à une liste triée, par dichotomie. Donner sa complexité dans le meilleur et le pire cas.
4. Écrire une fonction triant une liste par l'algorithme du tri rapide. Donner sa complexité dans le meilleur et le pire cas.
5. Écrire une fonction prenant en entrée un graphe représenté par listes d'adjacences, ainsi qu'un sommet de départ, et renvoyant la liste des sommets rencontrés dans un parcours en profondeur récursif depuis ce sommet de départ. Donner sa complexité dans le pire cas.

Exercice 4 : Programmation dynamique

On considère une grille de m lignes et n colonnes remplie d'entiers positifs, représentée par une liste de listes `G` où `G[i][j]` est la valeur de la case (i, j). On souhaite trouver le coût minimal d'un chemin allant de la case $(0, 0)$ à la case $(m-1, n-1)$, en ne se déplaçant que vers la droite ou vers le bas. Le coût d'un chemin est la somme des valeurs des cases traversées.

1. On note $d(i, j)$ le coût minimal pour atteindre la case (i, j) depuis $(0, 0)$. Donner la valeur de $d(0, 0)$ et établir la relation de récurrence vérifiée par $d(i, j)$.
2. Écrire une fonction `cout_min` qui prend en entrée la grille `G` et renvoie $d(m-1, n-1)$ par programmation dynamique ascendante.
3. Modifier la fonction pour qu'elle renvoie également le chemin optimal sous forme d'une liste de cases parcourues.