Devoir surveillé n° 2
Durée : 1h30min

Structure de données

Aucun document ni machine électronique n'est permis. Les questions peuvent être résolues de façon indépendante. La notation Ne tiendra pas compte des solutions bricolées et **l'utilisation de structures itératives ou récursives sans nécessite sera pénalisée**. La seule fonction préprogrammée des listes en langage Python autorisée est la fonction Len(), le reste elles doivent être créer avant de les utilisées.

Problème N°1 : *Distances et itinéraires*

On dispose d'informations sur 7 villes de France, classées dans les deux matrices données ci-dessous :

$$\texttt{infoVilles} = \begin{bmatrix} \text{"Marseille"} & \text{"13"} & \text{"5"} \\ \text{"Brest"} & \text{"29"} & \text{"2"} \\ \text{"Bordeaux"} & \text{"33"} & \text{"1"} \\ \text{"Tours"} & \text{"37"} & \text{"7"} \\ \text{"Grenoble"} & \text{"38"} & \text{"3"} \\ \text{"Lyon"} & \text{"69"} & \text{"4"} \\ \text{"Paris"} & \text{"75"} & \text{"6"} \end{bmatrix} \quad \texttt{distances} = \begin{bmatrix} 0 & 644 & 671 & 581 & 646 & 581 & 347 \\ 644 & 0 & 1110 & 1021 & 1284 & 597 & 500 \\ 671 & 1110 & 0 & 115 & 308 & 571 & 614 \\ 581 & 1021 & 115 & 0 & 317 & 465 & 524 \\ 646 & 1284 & 308 & 317 & 0 & 775 & 793 \\ 581 & 597 & 571 & 465 & 775 & 0 & 239 \\ 347 & 500 & 614 & 524 & 793 & 239 & 0 \end{bmatrix}$$

infoVilles est une matrice de **chaînes de caractères** à 7 lignes et 3 colonnes : pour chacune des 7 villes, la première colonne donne son **nom**, la seconde son numéro de département et la troisième son identifiant. Les villes sont classées par numéro de département croissant.

Par ailleurs, distances est une matrice de réels à 7 lignes et 7 colonnes, qui indique les distances (en km) entre toutes les paires de villes, classées selon leurs identifiants. Cette matrice est donc symétrique, et contient des 0 sur la diagonale.

Ainsi, pour connaître la distance entre Paris et Marseille, on déduit de la première matrice les identifiants correspondants, respectivement 6 et 5. On va donc lire dans la matrice distances la valeur contenue dans la ligne 6 colonne 5, et on trouve 775km.

L'objectif de l'exercice est d'utiliser ces matrices pour en extraire différentes informations. Vous pouvez utiliser dans une question les fonctions des questions précédentes (même si vous n'y avez pas répondu). La taille des deux matrices qui égale à 7 est à titre d'exemple le nombre de ville est plus grand que celui donné sur l'exemple et donc la taille 7 ne doit pas figurer dans les réponses.

Question 1 – Donner la fonction nomToId qui prend en argument une chaîne de caractères et a le comportement suivant :

- si la chaîne de caractères correspond à l'une des villes de la matrice infoVilles, la fonction renvoie **un entier** correspondant à l'identifiant de la ville.
- sinon, la fonction provoque une erreur

Ainsi nomToId("Paris") doit renvoyer l'entier 6, nomToId("Toulouse") doit provoquer une erreur.

Question 2 –Donner la fonction idToNom qui réalise le traitement inverse : étant donné un entier, la fonction renvoie le nom de la ville correspondante si l'identifiant est correct, elle renvoie une erreur sinon.

Ainsi idToNom(5) doit renvoyer la chaîne de caractères "Marseille".

Question 3 –Donner la fonction distanceEntre qui prend en argument deux chaînes de caractères et renvoie la distance qui sépare les deux villes correspondantes.

Ainsi distanceEntre("Paris","Marseille") doit renvoyer 775.

Question 4 – Donner la fonction distanceMax qui renvoie un triplé composé de la distance maximale observée, ainsi que les deux villes correspondantes.

Ainsi distanceMax() doit renvoyer la valeur 1284 et les chaînes de caractères "Brest" et "Marseille".

Question 5 – Donner la fonction distanceMin qui renvoie un triplé composé de la distance minimale (non nulle !) observée, ainsi que les deux villes correspondantes.

Ainsi distanceMin() doit renvoyer la valeur 115 et les chaînes de caractères "Lyon" et "Grenoble".

Question 6 –On s'intéresse à un itinéraire passant par plusieurs villes, dont on veut calculer la distance totale. Donner la fonction distanceItineraire qui prend en argument une liste contenant des chaînes de caractères (correspondant à des villes présentes dans la base), et qui calcule la distance totale du parcours réalisé en partant de la première ville et en allant de ville en ville jusqu'à la dernière.

Ainsi distanceItineraire(['Brest','Paris','Grenoble']) doit renvoyer 597 + 571 = 1168.

Question 7 –On s'intéresse au cas où` la matrice de distance est représentée sous forme concise, en supprimant les redondances et les 0, sous la forme d'un vecteur vecteurDistances : si on note v_1, ...v_7 les 7 villes considérées, ce vecteur compact contient les distances suivantes $d(v_1, v_2)$, $d(v_1, v_3)$, ..., $d(v_1, v_7)$, $d(v_2, v_3)$..., $d(v_2, v_7)$, ..., $d(v_6,v_7)$ Quelle est la taille de ce vecteur pour n villes ? (justifiez votre réponse)

Question 8 –Ecrire la fonction distanceEntreBis qui prend en argument deux chaînes de caractères et renvoie la distance qui sépare les deux villes correspondantes, en n'utilisant que le vecteur vecteurDistances déclaré en variable globale. Justifiez (commentez) votre réponse.

Devoir surveillé n° 2 Durée : 1h30min

Problème N °2 : *Jeu d'´echecs*

On s'intéresse dans cet exercice à quelques fonctions liées au jeu d'échecs. Il n'est pas nécessaire de savoir jouer aux échecs pour répondre aux questions.

On utilise la représentation suivante :

- un échiquier est représenté par une matrice de taille 8 × 8
- les pièces sont représentées par des entiers selon le codage suivant :

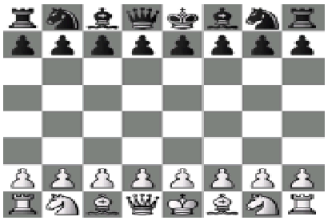
roi 1, reine 2, tour 3, fou 4, cavalier 5, pion 6

- les valeurs positives représentent des pièces blanches, les valeurs négatives des pièces noires, le 0 représente des cases vides.

Les deux matrices ci-dessous représentent deux exemples d'échiquiers : echiquier0 est le codage de l'état initial de l'échiquier, la matrice echiquierF représente la fin de la partie, où les blancs sont échec et mat.

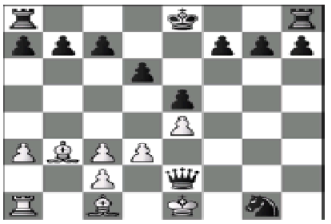
```
echiquier0 =
⎡ −3  −5  −4  −2  −1  −4  −5  −3 ⎤
⎢ −6  −6  −6  −6  −6  −6  −6  −6 ⎥
⎢  0   0   0   0   0   0   0   0 ⎥
⎢  0   0   0   0   0   0   0   0 ⎥
⎢  0   0   0   0   0   0   0   0 ⎥
⎢  0   0   0   0   0   0   0   0 ⎥
⎢  6   6   6   6   6   6   6   6 ⎥
⎣  3   5   4   2   1   4   5   3 ⎦
```

correspondant à



```
echiquierF =
⎡ −3   0   0   0  −1   0   0  −3 ⎤
⎢ −6  −6  −6   0   0  −6  −6  −6 ⎥
⎢  0   0   0  −6   0   0   0   0 ⎥
⎢  0   0   0   0  −6   0   0   0 ⎥
⎢  0   0   0   0   6   0   0   0 ⎥
⎢  6   4   6   6   0   0   0   0 ⎥
⎢  0   0   6   0  −2   0   0   0 ⎥
⎣  3   0   4   0   1   0  −5   0 ⎦
```

correspondant à



Le but de l'exercice est d'écrire des fonctions pour manipuler de tels échiquiers, en considérant deux types de fonction, qui concernent respectivement des opérations de base et le mouvement des pièces.

Opérations de base

Question 1 – Ecrire une fonction compteNbPiecesTotal qui prend en argument une matrice représentant un échiquier et qui renvoie le nombre de pièces présentes sur l'échiquier.

Ainsi compteNbPiecesTotal(echiquier0) doit renvoyer 32, compteNbPiecesTotal(echiquierF) doit renvoyer 22.

Question 2– Ecrire une fonction compteNbPieces qui prend en argument une matrice représentant un échiquier et qui renvoie, sous la forme de deux entiers, le nombre de pièces blanches et le nombre de pièces noires présentes.

Ainsi compteNbPieces(echiquier0) doit renvoyer les valeurs 16 et 16,
compteNbPieces(echiquierF) doit renvoyer les valeurs 9 et 13.

Question 3– Pour évaluer numériquement l'avantage respectif des deux joueurs, on peut

calculer un score pour chacun : pour cela, on considère que chaque pièce est associée à une valeur numérique, et on définit le score d'un joueur comme la somme des valeurs des pièces de sa couleur présentes sur l'échiquier.

Ecrire une fonction calculScore qui prend en argument une matrice représentant un échiquier et un vecteur représentant les valeurs des pièces et renvoie sous la forme de deux entiers, le score des blancs et le score des noirs.

Le vecteur des valeurs des pièces est un vecteur de taille 6 dont la i-ème composante contient le doublé : le numéro associé à la pièce et la valeur de la pièce.

Par exemple valeurPieces = [(1,100), (2,50),(3, 30) ,(4,20) (5,10),(6, 1)] indique que la valeur du roi, associé à l'entier 1, est 100

Ainsi calculeScore(echiquier0, [(1,100), (2,50),(3, 30) ,(4,20) (5,10),(6, 1)]) doit renvoyer les valeurs 278 et 278,

calculeScore(echiquierF, [(1,100), (2,50),(3, 30) ,(4,20) (5,10),(6, 1)]) doit renvoyer les valeurs 175 et 228.

Fonctions de mouvements

On souhaite maintenant écrire des fonctions permettant de déplacer les pièces sur l'échiquier. On définit pour cela une variable echiquier représentant un échiquier courant sur lequel on effectue les mouvements de pièces.

On suppose qu'il existe une fonction initialisation qui définit cette variable et l'initialise en lui donnant les valeurs indiquées dans la matrice echiquier0 précédente.

Question 4- Ecrire une fonction joueCoup, qui prend en argument un échiquier, les numéros de ligne i et colonne j de la position initiale et ceux de la position finale k et l du déplacement souhaité, et qui modifie l'échiquier en réalisant le déplacement indiqué.

NB : on fait l'hypothèse que les valeurs de i, j, k et l conviennent, et on ne fait donc pas de vérification. Ainsi, lors du premier coup, après l'appel joueCoup(echiquier,7, 4, 5, 4), qui correspond au déplacement d'un pion blanc, l'échiquier doit devenir

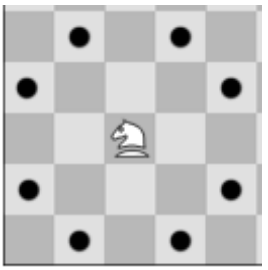
$$\begin{bmatrix} -3 & -5 & -4 & -1 & -2 & -4 & -5 & -3 \\ -6 & -6 & -6 & -6 & -6 & -6 & -6 & -6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 6 & 6 & \mathbf{0} & 6 & 6 & 6 & 6 \\ 3 & 5 & 4 & 2 & 1 & 4 & 5 & 3 \end{bmatrix}$$

Question 5 - On écrit ensuite des fonctions qui vérifient que les déplacements souhaités

conviennent. Ces fonctions dépendent des pièces considérées.

Ecrire une fonction verifieDeplacementCavalier qui prend en argument un échiquier, les numéros de lignes et colonnes des positions initiale et finale, i, j, k et l, souhaitées pour un cavalier et qui renvoie un booléen indiquant si ce déplacement est possible ou non. Pour cela, la fonction doit vérifier les points suivants :

- la case de départ doit contenir un cavalier,
- la case d'arrivée doit soit être vide, soit contenir une pièce de l'autre couleur,
- les positions initiale et finale doivent correspondre à un déplacement de cavalier.



On rappelle que le déplacement du cavalier consiste à aller de deux cases dans une direction (horizontale ou verticale), puis d'une case dans l'autre direction, comme illustré sur le graphe de la figure ci-contre. Ainsi, sur l'échiquier echiquier0 le cavalier blanc situé en (8,2) peut atteindre les cases (6,1) et (6,3). Déplacements possibles pour le cavalier

NB : L'échiquier est une matrice de 8x8, c'est-à-dire sont comprises entre 1 et 8,mais tous les structures de données sur Python commence l'indice à 0.