

TD/TP n° 2 : SQL et révisions de Python

1 Exercice d'algorithmique sur les listes

Dans cet exercice, on s'intéresse à une liste d'entiers de longueur n

$$L = [\ell_0, \ell_1, \dots, \ell_{n-1}].$$

Nous allons trier cette liste avec une stratégie d'énumération.

Par exemple, si $L = [4, 2, 8, 4, 5, 4, 2]$, on peut aisément calculer le minimum et le maximum de L : ce sont 2 et 8.

On peut alors dresser le tableau du nombre d'occurrences de chaque valeur dans L , comme suit. Ici, c_i est le nombre de fois où la valeur $\min(L) + i$ apparaît dans L .

i	0	1	2	3	4	5	6
$2 + i$	2	3	4	5	6	7	8
c_i	2	0	3	1	0	0	1

On peut alors trier L : on place d'abord les deux 2, puis les trois 4, puis le 5 et enfin le 8. Ainsi, la liste triée est

$$L' = [2, 2, 4, 4, 4, 5, 8].$$

- 1) Écrire une fonction `minmax` prenant en argument une telle liste L et renvoyant le couple (m, M) , où m est la valeur du minimum de L et M la valeur du maximum de L .
- 2) Écrire une fonction `compteurs` prenant en argument une telle liste L et renvoyant la liste C , de longueur $M - m + 1$, où pour tout $0 \leq i \leq M - m$, c_i est le nombre d'occurrences de l'entier $m + i$ dans L .

On fera attention à avoir une fonction de complexité linéaire, *i.e.* on évitera d'avoir deux boucles imbriquées l'une dans l'autre.

- 3) Écrire une fonction `tri` prenant en argument une telle liste L et renvoyant la liste triée.
- 4) En fonction de $M = \max(L)$, $m = \min(L)$ et de n (la taille de L), quelle est la complexité temporelle d'un appel de `tri(compteurs)` ? Et de `tri(L)` ?
- 5) Si l'on suppose que $M - m = O(n)$, comparer cette complexité à celle des tris vus l'année dernière.
- 6) Écrire une fonction `representation` prenant en argument une telle liste L et affichant le nuage des points (k, ℓ_k) .
- 7) On suppose que le contenu de L' est enregistré dans un fichier texte organisé comme suit : chaque ligne contient une valeur, puis le nombre d'occurrences de cette valeur (séparées par un espace), comme dans l'exemple suivant.

```
2 2
4 3
5 1
8 1
```

Écrire une fonction `lecture` prenant en argument le nom de ce fichier, et renvoyant la liste L' (triée).

2 Exercice sur la représentation des nombres

On donne le script suivant.

```
1 x1, x2 = 1., 1.+1e-15
2 a, b, c = 1., -(x1+x2), x1*x2
3 Delta = b**2-4*a*c
```

À la fin de l'exécution de ce script, la valeur affectée à la variable `Delta` est $-8.881784197001252e-16$. Un programme naïf déciderait donc que le trinôme $aX^2 + bX + c$ n'a pas de racine réelle.

- 1) Commenter de ce qui vient d'être décrit. Cela correspond-il à ce qui est prédit par le cours de mathématiques ? Pourquoi ?
- 2) *Facultatif* : Pouvait-on prévoir le signe de Δ donné ci-dessus ?
- 3) Donner le flottant représenté en hexadécimal par `'0x1.3a5f000000000p4'` (ne pas simplifier le calcul). On rappelle qu'en hexadécimal, $a = 10$, $b = 11$, $c = 12$, $d = 13$, $e = 14$ et $f = 15$.
- 4) On considère une image de 1920 pixels de largeur par 1080 pixels de hauteur (format 16 : 9). Chaque pixel est décrit par un mot de longueur 6 en hexadécimal. Par exemple, le mot `ff33e3` décrit un joli rose dont :
 - ▷ la composante rouge est décrite par `ff`, interprété comme $15 \times 16 + 15 = 255$;
 - ▷ la composante verte est décrite par `33`, interprété comme $3 \times 16 + 3 = 51$;
 - ▷ la composante bleue est décrite par `e3`, interprété comme $14 \times 16 + 3 = 227$.

Quelle est la taille en *Mo* (un *Mo* vaut 10^6 octets) occupée par cette image, si on la sauvegarde dans un format *bitmap*, en dehors de l'espace nécessaire à la structure de données elle-même ?

3 Exercice sur la notion de complexité et d'invariant

- 1) Donner la complexité de la fonction `produitMatrices` (rédiger en moins de 9 phrases).

```
4 def produitLigneColonne(M:list, N:list, i:int, j:int) -> float :
5     """Réalise le produit de la i-ème ligne de M par la j-ème
6         ↪ colonne de N
7     Précondition : M,N : matrices de dimensions compatibles"""
8     p = len(M[i])
9     S = 0
10    for k in range(p) :
11        S += M[i][k]*N[k][j]
12    return S
13
14 def produitMatrices(M:list, N:list) -> list :
15     """Renvoie le produit MN
16     Précondition : M,N : matrices de dimensions compatibles"""
17     n, m = len(M), len(N[0])
18     produit = [[0.]*m for i in range(n)]
19     for i in range(n) :
20         for j in range(m) :
21             produit[i][j] = produitLigneColonne(M, N, i, j)
22     return produit
```

- 2) On rappelle le principe de l'exponentiation rapide : pour toute matrice carrée M et tout entier $p \geq 2$:

$$M^p = \begin{cases} (M^{p/2})^2 & \text{si } p \text{ est pair,} \\ M(M^{p/2})^2 & \text{si } p \text{ est impair.} \end{cases}$$

Écrire une fonction récursive `puissance(M, p)` renvoyant la matrice M^p pour $p \geq 1$, en exploitant la relation précédente.

- 3) *Facultatif, difficile* : évaluer la complexité d'un appel de `puissance(M, p)` (on pourra considérer que M est de taille $n \times n$: chaque multiplication matricielle s'effectue en temps $O(n^3)$).
- 4) *Facultatif, et pour briller en société* : Calculer le n^{e} nombre de Fibonacci en temps logarithmique.
- 5) On donne le code suivant, où x est un flottant et n un entier naturel.

```
22 def puissance(x, n) :
23     a = 1
24     e = n
25     p = x
26     while e > 0 :
27         if e%2 == 1 :
28             a = a*p
29             p = p**2
30             e = e//2
31     return a
```

On va montrer que la relation $ap^e = x^n$ est un invariant de la boucle `while` de cette fonction.

- a) Justifier que la relation $ap^e = x^n$ est vraie à la fin de la ligne 25.
- b) On suppose que la relation $ap^e = x^n$ est vraie au début de la ligne 27.
En discutant en fonction de la parité de e , démontrer que cette relation est toujours vraie à la fin de la ligne 30.
- c) En sortie de la boucle `while`, que peut-on dire au sujet de e , vu que e est clairement une quantité entière et positive ?
- d) En déduire que cette fonction renvoie bien la valeur de x^n .
- 6) Démontrer que la quantité e est un variant de la boucle `while` de la fonction précédente.

4 Exercice d'algorithmique sur les matrices

On considère un robot se déplaçant sur un plateau rectangulaire, de n cases de haut et de p cases de large. Le robot se situe initialement dans le coin supérieur gauche du plateau, correspondant à la ligne $n^{\circ} 0$ et la colonne $n^{\circ} 0$.

Le robot peut se déplacer dans les quatre directions : vers le haut, vers le bas, vers la droite et vers la gauche.

Sur certaines cases du plateau sont disposés des obstacles. Ces obstacles sont représentés par une matrice booléenne `libre` : pour tout $0 \leq i < n$ et $0 \leq j < p$, `libre[i][j]` vaut `True` si la case située sur la ligne $n^{\circ} i$ et la colonne $n^{\circ} j$ ne possède pas d'obstacle, et `False` sinon.

Enfin, on donne une liste `deplacements` de déplacements au robot.

On a représenté dans la figure 1 un tel exemple de grille ($n = 3$ et $p = 4$). La matrice d'obstacles serait alors

```
libre = [[True, True, False, True],
         [True, True, False, True],
         [False, True, True, True]]
```

et la liste de déplacements

```
deplacements = ["Droite", "Bas", "Bas", "Droite", "Droite", "Haut"]
```

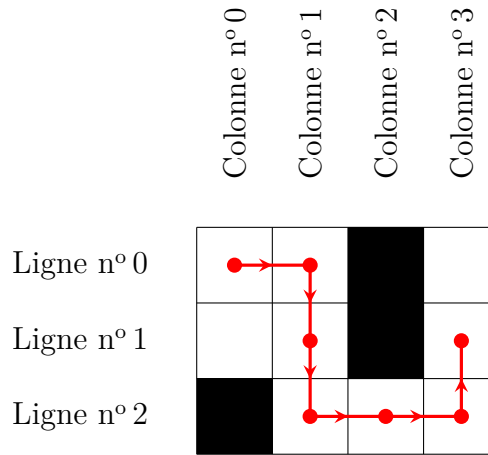


FIGURE 1 – Exemple d'obstacles et de déplacement du robot ($n = 3$, $p = 4$)

Écrire une fonction **arrivee** prenant en argument la matrice **libre** et la liste **deplacements** et qui renvoie :

- ▷ $(-1, -1)$ si le robot sort du plateau ou rencontre un obstacle ;
- ▷ les coordonnées de la case d'arrivée du robot sinon, sous la forme d'un couple (numéro de ligne, numéro de colonne).

5 Exercice sur les bases de données

Nous étudions dans cet exercice la base de données d'une plateforme (fictive) de jeux-vidéos. Cette plateforme permet à des personnes de jouer à distance à des jeux. Elle contient trois tables, que l'on décrit ci-dessous.

- Table **jeux**, qui contient les attributs suivantes :
 - **id** : numéro d'identification du jeu ;
 - **titre** : titre du jeu ;
 - **sortie** : date de sortie du jeu.
- Table **joueurs**, qui contient les attributs suivantes :
 - **pseudo** : pseudonyme du joueur ;
 - **email** : adresse électronique du joueur ;
 - **date_naissance** : date de naissance du joueur.
- Table **parties**, qui contient les attributs suivantes :
 - **id** : numéro d'identification de la partie ;
 - **idjeu** : numéro d'identification du jeu ;
 - **pseudo** : pseudonyme du joueur ;
 - **date** : date de la partie ;
 - **duree** : durée de la partie (en secondes) ;
 - **langue** : langue dans laquelle a été jouée la partie.

Une annexe se trouve en fin de sujet. Il s'agit d'exemples dont l'objectif est d'aider à la compréhension de la base de données. Leur contenu ne doit pas être utilisé pour répondre aux questions !

Les réponses des questions **2) à 15)** seront données sous forme de requêtes SQL.

- 1) Dessiner un schéma entité-association de la base de données. Spécifier les clefs primaires des entités, et les clefs étrangères présentes dans chaque table.
- 2) Obtenir l'entièreté de la table **jeux**.
- 3) Obtenir la liste des titres de tous les jeux.
- 4) Obtenir la liste des pseudonymes et des adresses électroniques des joueurs nés en 1988.
- 5) Obtenir le nombre de joueurs nés le 29 février 1988.
- 6) Obtenir la liste des titres des jeux auxquels a joué le joueur de pseudonyme **PrShadoko**.
- 7) Obtenir la liste des cinq derniers jeux auxquels a joué le joueur de pseudonyme **PrShadoko**.
- 8) Obtenir la liste des titres en doublons dans le catalogue de jeux de la plateforme.
- 9) Obtenir le temps de jeu total sur **Rebellion** du joueur de pseudonyme **PrShadoko**, en secondes.
- 10) Obtenir le temps de jeu total sur **Rebellion** du joueur de pseudonyme **PrShadoko**, en heures.
- 11) Obtenir la durée de jeu total sur chaque jeu du joueur de pseudonyme **PrShadoko**, en heures.
- 12) Obtenir la liste des jeux auxquels le joueur de pseudonyme **PrShadoko** a joué au moins 10 heures.
- 13) Obtenir le titre du ou des jeux dont la durée moyenne de jeu par joueur est la plus élevée.
- 14) Obtenir les titres des jeux qui n'ont pas été joués par le joueur dont le pseudonyme est **PrShadoko**.
- 15) Obtenir les titres des jeux qui ont été joués à la fois en **français** et en **klinton**.

Annexe

Les tableaux ci-dessous sont donnés à titre d'**exemples** ; ils contiennent des **extraits** des trois tables. Leur contenu ne doit donc pas être utilisé pour répondre aux questions.

Extrait de la table jeux

id	titre	sortie
...	...	...
27	"Monkey Island"	"1990-10-15"
28	"Rebellion"	"1998-03-24"
29	"Myst"	"1993-09-24"
...	...	...

Extrait de la table joueurs

pseudo	email	date_naissance
...	...	...
"DevinPlombier"	"devin@planeteshadok.sh"	"1981-12-21"
"Gibi"	"gibi@planeteshadok.sh"	"1988-03-20"
"Tux"	"logiciellibre@linux.os"	"1992-05-12"
...	...	...

Extrait de la table parties

id	idjeu	pseudo	date	duree	langue
...	...	...	...	...	...
215421	16	"Gibi"	"2023-05-17"	3852	"français"
215422	27	"PrShadoko"	"2022-12-25"	4721	"anglais"
215423	49	"Corwin"	"2021-01-13"	1237	"klingon"
...	...	...	...	...	...