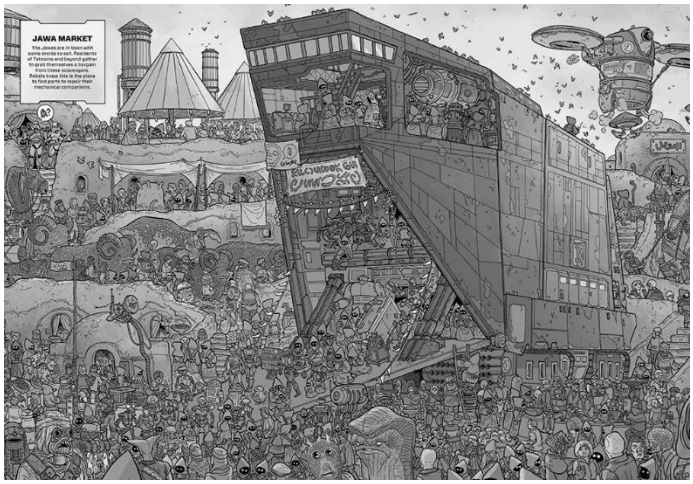
Terminale NSI

Bac Blanc 2023

By DarkSATHI Li

Si c'est trop difficile ou trop facile….Cherche après Chewbacca et dis-lui de rentrer à la maison !!!!

Les interdictions



Exercice 1 : 5 points

Un Sudoku classique est une grille de 9 lignes et 9 colonnes, donc composée de 9 blocs distincts de 3 lignes et 3 colonnes.
Remplir une grille de Sudoku consiste

- à utiliser tous les chiffres de 1 à 9 pour chacun des 9 blocs;
- chaque ligne et chaque colonne de la grille totale ne comporte aucun chiffre en double.

Dans cet exercice, on décidera de représenter une grille de Sudoku(`sudo`) par une liste de neuf listes à neuf éléments. Chaque liste représente une ligne du jeu. Le chiffre 0 marque l'absence de remplissage de la case.

	5		8		7		2	
6				1			9	
7		2	5	4				6
	7			2		3		1
5		4				9		8
1		3		8			7	
9				7	6	2		5
	6			9				3
	8		1		3		4	

Figure 1

1. On donne `sudo = [[0, 5, 0, 8, 0, 7, 0, 2, 0], ...]`.
   Donner la liste d'indice 4 de la liste `sudo` représentant le sudoku de la figure 1.
2. Ecrire une fonction `ligne(sudo, i)`, en langage Python, qui retourne la liste des chiffres de 1 à 9 qui apparaissent sur la ligne d'indice `i`.

   Exemple :

   ```
   >>> ligne(sudo, 0)
   [5, 8, 7, 2]
   ```

3. Recopier et compléter la fonction `colonne(sudo, j)`, en langage Python, qui retourne la liste des chiffres de 1 à 9 qui apparaissent dans la colonne `j`.

(Les lignes 2, 3, 4, 5 et 6)

```
1 def colonne(sudo, j):
2     tableColonne = ....
3     for i in range(....):
4         if sudo[i][j] .... 0:
5             tableColonne.append(....)
6     return ....
```

4. On veut écrire une fonction `bloc(sudo, i, j), en langage Python`, qui retourne la liste des chiffres de 1 à 9 qui apparaissent dans le bloc 3×3 auquel appartient l'élément qui se trouve à l'intersection de la ligne d'indice i et de la colonne d'indice j.
   a. Recopier la proposition qui permet d'obtenir l'indice de la ligne `k` et l'indice de la colonne `l` de l'élément qui se situe en haut et à gauche du bloc auquel appartient l'élément qui se trouve à l'intersection de la ligne d'indice `i` et de la colonne d'indice `j`.

      - `k = i - i//3 et l = j - j//3`
      - `k = i - i%3 et l = j - j%3`
      - `k = i – 3 et l = j - 3`
      - `k = i et l = j`

   b. On donne la ligne `k` et la colonne `l` de la case se trouvant en haut à gauche d'un bloc. Compléter la fonction `bloc_init(k, l)` qui renvoie la liste des chiffres de 1 à 9 qui apparaissent dans le bloc 3×3 auquel appartient la case de la ligne `k` et de la colonne `l`.

      (Les lignes 8, 9, 11 et 12)

```
7   def bloc_init(sudo, k, l):
8       listeBloc = ....
9       for m in range(....):
10          for n in range(3):
11              if sudo([k + m][l + n] != ....
12                  .....................
13      return listeBloc
```

   c. En déduire alors une écriture possible de la fonction `bloc(sudo, i, j)`.

5. Ecrire alors une fonction `chiffres_interdits(sudo, i, j)` qui retourne la liste des chiffres de 1 à 9 que l'on ne peut pas écrire dans la case de la ligne `i` et de la colonne `j`, en suivant les règles du SUDOKU.

   Exemple :

```
>>> chiffres_interdits(sudo, 0, 0)
[5, 8, 7, 2, 6, 1, 9]
```

6. Ecrire une fonction `suivante(i, j)` qui reçoit en paramètre la ligne `i` et la colonne `j` d'une case, et qui renvoie la tuple (ligne, colonne) de la case suivante. On renverra sans se poser de question (9, 0) lorsque la fonction est appelée avec les paramètres `i=8` et `j=8`.

   Exemple:

```
>>> suivant(0, 0)
(0, 1)
>>> suivante(0, 8)
(1, 0)
>>> suivante(8, 8) # Attention!!
(9, 0)
```

Le retour sur trace ou retour arrière (appelé aussi backtracking en anglais) est une famille d'algorithmes pour résoudre des problèmes algorithmiques, notamment de satisfaction de contraintes (optimisation ou décision). Ces algorithmes permettent de tester systématiquement l'ensemble des affectations potentielles du problème. Ils consistent à sélectionner une variable du problème, et pour chaque affectation possible de cette variable, à tester récursivement si une solution valide peut-être construite à partir de cette affectation partielle. Si aucune solution n'est trouvée, la méthode abandonne et revient sur les affectations qui auraient été faites précédemment (d'où le nom de retour sur trace).
En d'autres termes, le retour sur trace est un parcours en profondeur sur l'arbre de décision du problème.

La fonction récursive `solve(i, j)` doit renvoyer `True` si l'on a réussi à compléter toute la grille à partir des hypothèses faites dans les cases précédant `(i, j)` (pour l'ordre de la question précédente) et `False` dans le cas contraire.

Le principe est le suivant :

- Le cas de base est lorsqu'on a réussi à remplir la grille : on a `i = 9` et `j = 0`. Dans ce cas on renvoie `True`.
- sinon si la case `sudo[i][j]` est déjà remplie (c'était une des données du Sudoku), On doit mettre à jour deux variables `a` et `b` pour passer à la case suivante.
- sinon, on calcule les chiffres que l'on ne peut pas mettre en case `(i, j)`, et on essaie successivement tous les autres : on écrit un chiffre en case `(i, j)` et on passe en case suivante par appel récursif. Il est nécessaire de récupérer le booléen associé à un tel appel.
    - s'il est True on a réussi à remplir la grille à partir des suppositions sur les cases précédentes,
    - s'il est False c'est que l'essai n'est pas concluant.

7.

a. Compléter la fonction principale `solve(sudo, i, j)` qui permet de résoudre le SUDOKU par backtracking(si cela est possible).

(Les lignes 15, 16, 19, 23, 24, 26 et 28)

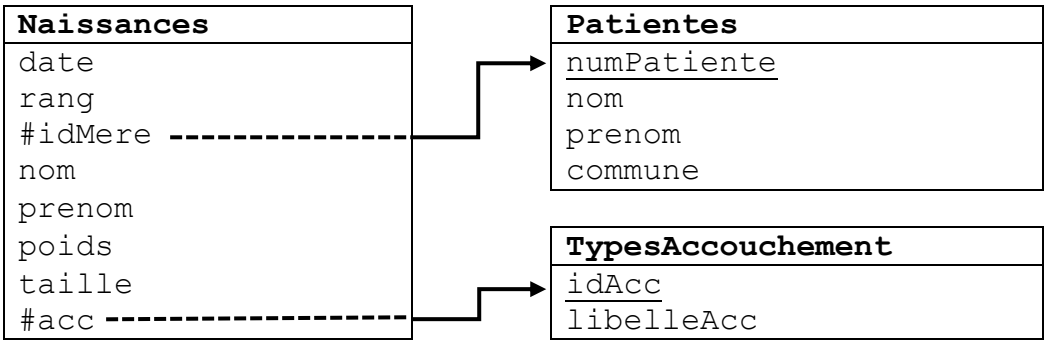
```
14 def solve(i, j):
15     if i == ...:
16         return ...
17     elif sudo[i][j] > 0:
18         a, b = suivante(i, j)
19         return ...(a, b)
20     else :
21         a, b = suivante(i, j)
22         for entier in range(1, 10):
23             if entier not in ..........................:
24                 sudo[i][j] = ......
25                 if solve(a, b):
26                     return ...
27     sudo[i][j] = 0
28     return ...
```

b. Préciser les valeurs à donner aux paramètres `i` et `j` pour que l'appel `solve(i, j)` donne la solution du SUDOKU(Si elle existe).

Exercice 2 : 4 points

Le service maternité d'un hôpital utilise une base de données pour gérer les informations concernant les naissances qui y ont lieu.

Le schéma relationnel de cette base de données est le suivant.



Dans la relation `Naissances` :

- La clé étrangère `idMere` fait référence à la clé primaire `numPatiente` de la relation `Patientes`.
- La clé étrangère `acc` fait référence à la clé primaire `idAcc` de la relation `TypesAccouchement`. Il permet de savoir si la patiente a accouché par voie naturelle ou par césarienne.
- L'attribut `rang` indique le rang de naissance du bébé dans le mois. Il recommence donc à 1 au début de chaque nouveau mois.
- L'attribut `poids` est exprimé en grammes et l'attribut `taille` en centimètres.

Dans cet exercice, on pourra utiliser les mots clés suivants du langage SQL :
`SELECT, DELETE, FROM, WHERE , JOIN, INSERT,UPDATE, MIN, MAX, AVG.`

Les fonctions d'agrégation `MIN(attribut)`, `Max(attribut)` et `AVG(attribut)` renvoient respectivement la plus petite valeur, la plus grande et la valeur moyenne de l'attribut `attribut` pour les enregistrements sélectionnés. Ainsi la requête `SELECT MIN(taille) FROM Naissances` renvoie la plus petite valeur de l'attribut `taille` de la table `Naissances`.

On donne ci-après des extraits des tables renvoyées par certaines requêtes SQL.

```
SELECT * FROM Naissances;
```

date	rang	idMere	nom	prenom	poids	taille	acc
…	…	…	…	…	…	…	…
28/02/2022	263	13 857	Berthelot	Maïssa	3305	50	1
28/02/2022	264	13 858	Samson	Pauline	3650	52	1
28/02/2022	265	13 859	Perrin	Jonathan	3720	52	2
01/03/2022	1	13 860	Fernandez	Lorette	3350	51	2
01/03/2022	2	13 861	Baugé	Juliette	2870	48	1
01/03/2022	3	13 861	Baugé	Noé	2985	49	1

`SELECT * FROM Patientes;`

numPatiente	nom	prenom	commune
...	...	...	...
13 857	Berthelot	Michelle	Aigrefeuille d'Aunis
13 858	Samson	Marine	Nieul sur Mer
13 859	Perrin	Patricia	La Rochelle
13 860	Fernandez	Claire	Theed
13 861	Baugé	Gaëlle	Lagord

`SELECT libelleAcc FROM TypesAccouchement;`

LibelleAcc
voie naturelle
césarienne

Attention, la table est complète.

1. Donner la définition d'une clé primaire.
2. Donner la définition d'une clé étrangère.
3. Proposer un couple d'attributs qui peut être une clé primaire pour la relation `Naissances` sans ajouter de nouvel attribut à cette relation.
4. Expliquer pourquoi la requête ci-dessous va provoquer une erreur.

```
DELETE FROM Patientes WHERE nom="Baugé" and prenom="Gaëlle" ;
```

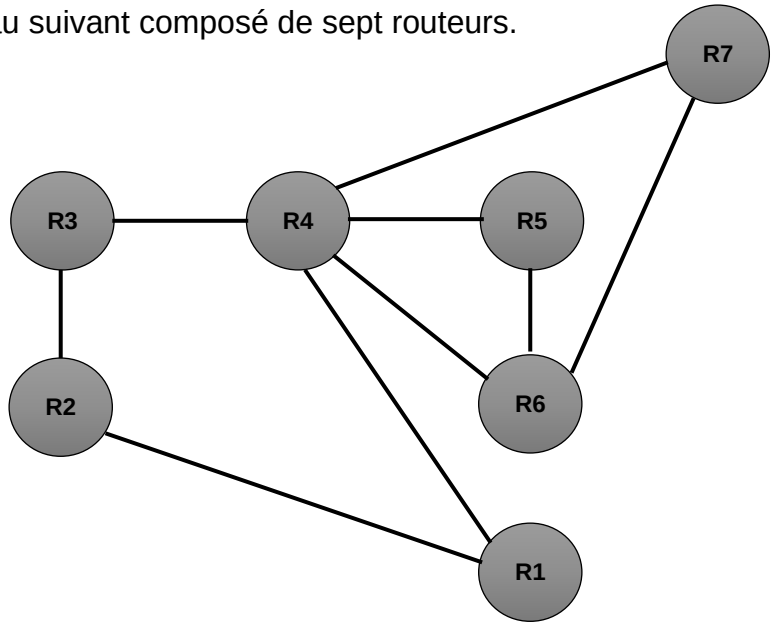
5. Donner en langage SQL la requête d'insertion qui permet d'inscrire, avec le numéro 13862, la patiente dénommée Ninette Bélanger résidant à la Rochelle.
6. Donner en langage SQL les requêtes d'insertion qui permettent d'inscrire, avec le numéro 13863, la patiente dénommée Padmé Amidala résidant à Theed(Capitale humaine de Naboo). Padmé a donné naissance, par accouchement physiologique(`idAcc=3`), le 22 janvier 9972, à des jumeaux Luke Skywalker(`rang=1`) et Leia Skywalker(`rang=2`) de poids et taille identiques(3000 grammes et 58 cm).
7. Expliquer pourquoi il serait judicieux que la table `Patientes` soit divisée en deux tables. Modifier alors le schéma relationnel donné en début d'exercice.(On utilisera pas la réponse à cette question dans la suite de l'exercice.)
8. Dans l'extrait de la table fourni, on constate qu'une erreur a été commise lors de la saisie du prénom du bébé de madame Fernandez. La bonne orthographe est Laurette. Donner en langage SQL l'instruction à exécuter pour corriger le prénom du bébé.
9. Ecrire une requête SQL qui renvoie la liste des patientes(`nom` et `prenom`) dont la commune de résidence est Theed.
10. Ecrire une requête SQL qui renvoie le poids moyen des bébés nés par voie naturelle.
11. Donner les enregistrements renvoyés par la requête suivante lancée sur les extraits donnés des tables.

```
SELECT Patientes.nom, Patientes.prenom FROM Naissances
     JOIN TypesAccouchement
     ON Naissances.acc=TypesAccouchement.idAcc
     JOIN Patientes
     ON Naissances.idMere=Patientes.numPatiente
WHERE TypesAccouchement.idAcc=1;
```

Exercice 2 : 3 points

Cet exercice porte sur les réseaux et les protocoles de routage.

On considère le réseau suivant composé de sept routeurs.



On donne les tables de routage préalablement construites ci-dessous avec le protocole RIP. Le protocole RIP permet de construire les tables de routage des différents routeurs, en indiquant pour chaque routeur, la distance, en nombre de sauts, qui le sépare d'une autre routeur.

Table de routage de R1		
Destination	Lien	Distance
R2	R2	1
R3	R4	2
R4	R4	1
R5	R4	2
R6	R4	2
R7	R4	2

Table de routage de R2		
Destination	Lien	Distance
R1	R1	1
R3	R3	1
R4	R1	2
R5	R3	3
R6	R3	3
R7	R1	3

Table de routage de R3		
Destination	Lien	Distance
R1	R2	2
R2	R2	1
R4	R4	1
R5	R4	2
R6	R4	2
R7	R4	2

Table de routage de R4		
Destination	Lien	Distance
R1	R1	1
R2	R3	2
R3	R3	1
R5	R5	1
R6	R6	1
R7	R7	1

Table de routage de R5		
Destination	Lien	Distance
R1	R4	2
R2	R4	3
R3	R4	2
R4	R4	1
R6	R6	1
R7	R6	2

Table de routage de R6		
Destination	Lien	Distance
R1	R4	2
R2	R4	3
R3	R4	2
R4	R4	1
R5	R5	1
R7	R7	1

Table de routage de R7		
Destination	Lien	Distance
R1	R4	2
R2	R4	3
R3	R4	2
R4	R4	1
R5	R4	2
R6	R6	1

1. Le routeur R2 doit envoyer un paquet de données au routeur R7 qui en accuse réception. Déterminer le chemin parcouru par le paquet de données ainsi que celui parcouru par l'accusé de réception.
2.
    a. Indiquer la faiblesse que présente ce réseau en cas de panne du routeur R4.
    b. Proposer une solution pour y remédier.
3. Dans cette question uniquement, on décide de rajouter un routeur R8 qui sera relié aux routeurs R2 et R6.
    a. Donner une table de routage pour R8 qui minimise le nombre de saut.
    b. Donner une nouvelle table de routage de R2.
4. Pour la suite de l'exercice on considèrera le réseau sans le routeur R8.
   Il a été décidé de modifier les règles de routage de ce réseau en appliquant dorénavant Le protocole de routage OSPF qui prend en compte la bande passante.
   Ce protocole attribue un coût à chaque liaison afin de privilégier le choix de certaines routes plus rapides. Plus le coût est faible, plus le lien est intéressant.
   Le coût d'une liaison est calculé par la formule :

$$coût = \frac{10^8\ bit/s}{bande\ passante\ du\ lien\ en\ bit/s}$$

   Voici le tableau référençant les coût des liaisons en fonction du type de liaison entre deux routeurs :

Type de liaison	Bande passante	Coût
FastEthernet (FE)		1
Ethernet (E)	10 Mb/s	
(E1)	2,048 Mb/s	49
(T1)	1,544 Mb/s	65

   On rappelle que $1\ Mb/s\ =\ 1000\ kb/s\ =\ 10^6\ bit/s$.

    a. Déterminer la bande passante du FastEthernet (FE) et justifier que le coût du réseau de type Ethernet est de 10.
    b. On considère sur le graphe ci-dessous les types de liaison dans le réseau . Donner le coût du chemin reliant R2 à R5 si le protocole de routage utilisé est OSPF.