Index des sujets 2022

22-NSIJ1AS1 : Corrigé

Année : **2022**
Centre : **Amérique du sud**
Jour : **1**
Enoncé :

1. Exercice 1

bases de données

1. Une clé primaire doit être unique pour chaque enregistrement, donc :
   - `id_mere` ne peut pas servir de clé primaire pusiqu'une même femme peut avoir plusieurs enfants, par exemple dans l'extrait de table fourni, la mère d'`idMere` 13861 apparaît deux fois.
   - `(date, rang)` peut servir de clé primaire, en effet pour un jour donné, le rang de naissance est unique.
   - `(poids,taille)` ne peut pas servir de clé primaire puisque deux bébés différents peuvent être nés avec le même poids et la même taille.
2. Une clé étrangère doit être toujours présente en tant que clé primaire dans la table qu'elle référence. Ici, la clé étrangère `idMere` doit donc être présente en tant que clé primaire dans la table `Patientes`. La requête

   🗂️ **Requête SQL**

   ```
   DELETE FROM Patientes WHERE numPatiente = 13858;
   ```

   produit donc une erreur, car dans la table `Naissances`, un enregistrement ayant pour `idMere` la valeur `13858` existe.

> **Note**
>
> De manière moins formelle, dans ce schéma de base de données, un bébé (enregistrement de `Naissances`) à nécessairement une mère (enregistrement de `Patientes`). On ne peut donc pas supprimer un enregistrement de `Patientes` qui correspond à une naissance.

3. 🗂️ **Requête SQL**

```
INSERT INTO Patientes VALUES(13862,"Bélanger","Ninette","La Rochelle");
```

4. 🗂️ **Requête SQL**

```
UPDATE Naissances SET prenom = "Laurette"
WHERE date = "01/03/2022" and rang = 1;
```

> **Note**
>
> On modifie le prénom en sélectionnant le bébé par sa date et son rang de naissance (qui peut servir de clé primaire d'après la question 1).

5. 🗂️ **Requête SQL**

```
SELECT nom,prenoms FROM Patientes
WHERE commune = "Aigrefeuille d'Aunis"
```

6. 🗂️ **Requête SQL**

```
SELECT AVG(poids) FROM Naissances
JOIN TypesAccouchement ON TypesAccouchement.idAcc = Naissances.acc
WHERE TypesAccouchement.libelleAcc = "césarienne"
```

7. Cette requête renvoie :

nom	prenom
Berthelot	Michelle
Samson	Marine
Baugé	Gaëlle

C'est à dire les noms et prénoms des patientes ayant eu un accouchement de type 1.

2. Exercice 2

programmation et algorithmes de tri

Note

Un patient `p` est représenté par un tuple contenant son identifiant et son ordre de priorité. Donc `p[0]` est l'identifiant et `p[1]` la priorité.

1. 🐍 **Script Python**

```
attente.append((50,4))
```

2. a. C'est le *tri par sélection* (à chaque passage dans la boucle `for i in range(len(attente)` on recherche le patient le plus prioritaire à partir du ième et on le place en position `i`)

Bug

la fonction `tri` modifie en place la liste `attente` mais ne renvoie rien contrairement à ce qui est indiqué dans l'énoncé

b. La complexité en temps des tris par insertion et par sélection est **quadratique** : \(\mathcal{O}(n^2)\).

3. a.

🐍 **Script Python**

```
def quitte(attente):
    return [patient from attente if patient[1]!=1]
```

b.

🐍 **Script Python**

```
def maj(attente):
    return [(patient[0],patient[1]-1) for patient in attente]
```

4. a.

🐍 **Script Python**

```
def priorite(attente,p):
    for patient in attente:
        if patient[0]==p:
            return patient[1]
```

b.

```
23    def revise(attente,p):
24        nouvelle = []
25        n = priorite(attente,p)
26        for (patient,prio) in attente:
27            if patient == p :
28                nouvelle.append((patient,1))
29            elif prio < n:
30                nouvelle.append((patient,prio+1))
31            else:
32                nouvelle.append((patient,prio))
33        return nouvelle
```

3. Exercice 3

arbres binaires

Attention

- L'implémentation des arbres binaires sous forme de dictionnaire telle que donné dans l'énoncé est *inhabituelle*.
- La fonction de parcours produit un affichage à l'aide d'instructions `print` mais ne renvoie rien.

1. On rappelle que la taille est le nombre de noeuds et la hauteur le nombre maximal de noeuds d'une branche. Dans le cas de l'arbre donné en introduction, la taille est donc **11** et la hauteur **5**.
2. a. Cette structure correspond à l'arbre 2. En effet, le noeud `d` est le sous arbre droit de `b` et le noeud `g` est le sous arbre gauche de `f`.

   b.

```
graph TD
H["H"] --> G["G"]
H --> F["F"]
G --> E["E"]
G --> D["D"]
D --> V1[" "]
D --> B["B"]
F --> C["C"]
F --> V2[" "]
C --> V3[" "]
C --> A["A"]
style V1 fill:#FFFFFF, stroke:#FFFFFF
style V2 fill:#FFFFFF, stroke:#FFFFFF
style V3 fill:#FFFFFF, stroke:#FFFFFF
linkStyle 4 stroke:#FFFFFF,stroke-width:0px
linkStyle 7 stroke:#FFFFFF,stroke-width:0px
linkStyle 8 stroke:#FFFFFF,stroke-width:0px
```

3. a. L'affichage obtenu sera : `d, b, g, f, a` (on reconnaît un parcours en profondeur suffixé c'est à dire qu'on liste les noeuds du sous arbre gauche et du sous arbre droit avant la racine)

   b. On reprend la fonction `parcours` donnée dans l'énoncé, et on n'affiche l'étiquette seulement si le noeud est une feuille c'est à dire lorsque `arb['sag']` et `arb['sad']` sont l'arbre vide.

```
def parcours(arb):
    if arb == {}:
        return None
    parcours(arb['sag'])
    parcours(arb['sad'])
    if arb['sag'] == {} and abr['sad'] == {}:
        print(arb['etiquette'])
```

4.

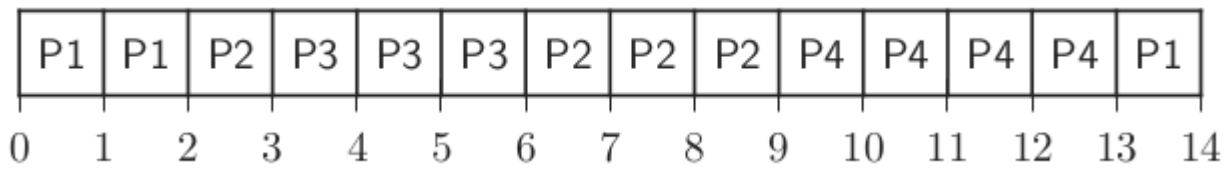
```
    def symptomes(arb,mal):
        if arb['sag'] != {}:
            symptomes(arb['sag'],mal)

        if arb['sag'] != {}:
            symptomes(arb['sad'],mal)

        if arb['etiquette']==mal:
            arb['surChemin'] = True
            print('symptômes de', abr['etiquette'], ':')

        else:
            if abr['sad'] != {} and abr['sad']['surChemin'] :
                print(abr['sad']['etiquette'])
                arb['surChemin'] = True

            if abr['sag'] != {} and abr['sag']['surChemin'] :
                print('pas de', abr['sad']['etiquette'] )
                arb['surChemin'] = True
```

4. Exercice 4

gestion des processus et des ressources par un système d'exploitation

Partie A : Ordonnancement des processus

1.



2.

Processus	Temps d'exécution	Instant d'arrivée	Temps de séjour	Temps d'attente
P1	\(3\)	\(0\)	\(14-0=14\)	\(14 - 3 = 11\)
P2	\(4\)	\(2\)	\(9-2=7\)	\(7 - 4 = 3\)
P3	\(3\)	\(3\)	\(6-3=3\)	\(3 - 3 = 0\)
P4	\(4\)	\(5\)	\(13-5=8\)	\(8 - 4 = 4\)

3. Le temps d'attente d'un processus est nul lorsque le temps de séjour est égal au temps d'exécution. C'est à dire lorsque le processus était le plus prioritaire durant la totalité de son temps d'exécution.

Partie B : Processus et ressources

1. D'après le tableau on constate que :
   - l'analyseur d'échantillon attend la donnée D4 qu'il ne peut obtenir car elle est mobilisée par le sgbd.
   - le sgbd attend la donnée D5 qu'il ne peut obtenir car elle est mobilisée par la tableur.
   - le tableur attend la donnée D1 qu'il ne peut obtenir car elle est mobilisée par l'analyseur d'échantillon.
2. C'est une situation d'interblocage ou *deadlock* en anglais.
3. Si D1 est libérée, alors le tableur peut s'exécuter, il libérera alors D3 ce qui permet l'exécution du sgbd et D5 qui permet l'exécution du traitement de texte. Un ordre possible d'exécution des processus est donc : tableur > sgbd > traitement de texte > analyseur d'échantillon.

5. Exercice 5

réseaux et protocoles de routage

Partie A : Adressage

1. L'adresse du service de radiologie est `192.168.1.0/24` c'est à dire que l'adresse du réseau est `192.168.1.0` et le masque `255.255.255.0`
2. Les adresses des trois interfaces du routeur R5 sont :
   - `172.89.50.0/24`
   - `44.197.5.0/24`
   - `192.168.5.0/24`

3. a. La première adresse pouvant être attribuée à une machine sur le réseau RL R est `192.168.1.1` et la dernière `192.168.1.254`. b. Un maximum de **254** machines peuvent donc être connectées sur ce réseau.

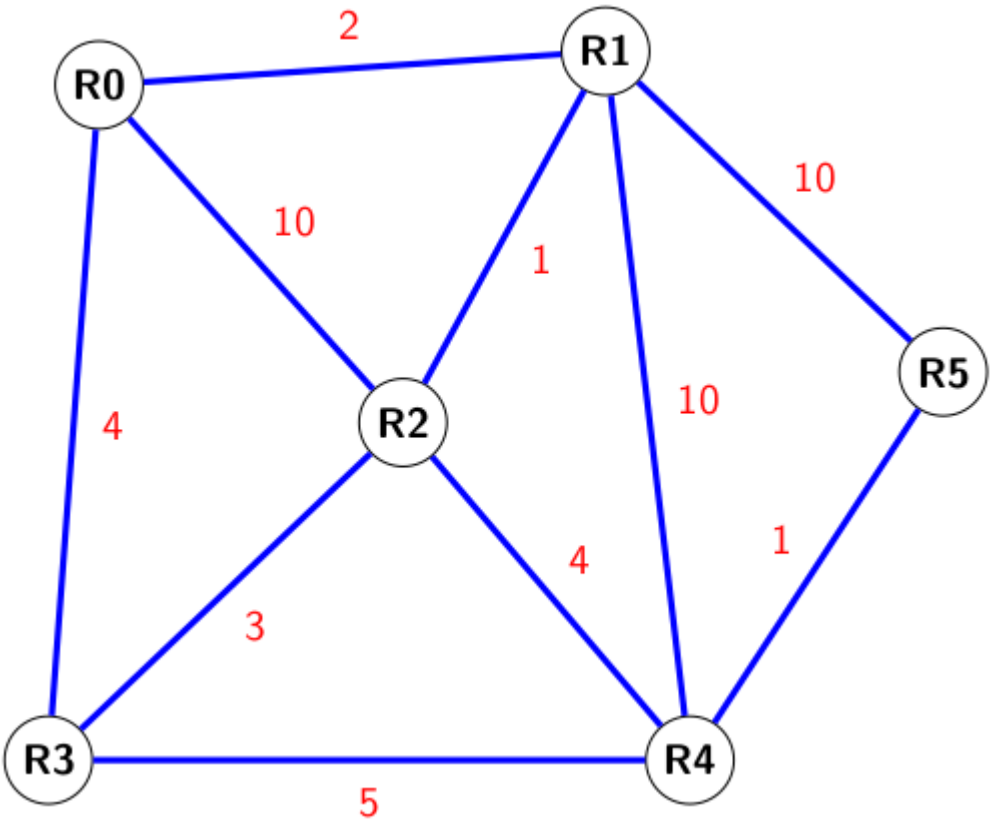
Partie B : Etude du protocole RIP

1. Les routeurs parcourus seront `R5`
   →
   `R1`
   →
   `R0`
2. En cas de panne du routeur `R1`, une nouvelle route sera : `R5`
   →
   `R4`
   →
   `R2`
   →
   `R0`

Partie B : Etude du protocole OSPF

> **Bug**
>
> L'exemple de calcul du coût donné dans l'énoncé ne correspond pas aux valeurs des tableaux. Entre `R0` et `R3`, le coût devrait être calculé avec \(\dfrac{10^9}{300\times10^6} \simeq 3,33\) et l'entier immédiatement supérieur est 4. L'exemple donné semble correspondre au calcul du coût entre R1 et R2.

1. Le coût de la liaison entre `R2` et `R3` est : \(\dfrac{10^9}{400\times10^6}=2,5\). Le coût est l'entier immédiatement supérieur donc 3.
2. Une bande passante possible de la connexion entre `R3` et `R4` est \(200\) MB/s, en effet le coût sera alors \(\dfrac{10^9}{200\times10^6}=5\).
3. On a reproduit le graphe du réseau ci-dessous en faisant figurer les coûts des liaisons :



Le chemin parcouru sera donc `R0`

→

`R1`

→

`R2`

→

`R4`

→

`R5` et le coût total de ce chemin est \(2+1+4+1=8\).

4. En cas de panne du routeur `R1`, la nouvelle route sera : `R0`

→

`R3`

→

`R4`

→

`R5` et le coût total sera : \(4+5+1=10\)