Baccalauréat Général

Session 2024

Épreuve : **Numérique et sciences de l'ingénieur**

Durée de l'épreuve : 3h30

Coefficient : 16

PROPOSITION DE CORRIGÉ

Exercice 1 :

1) Le site2 n'a pas de site qui renvoie vers lui, donc il n'a pas de prédécesseurs. La liste s2.predecesseurs est donc représentée par une liste vide [].

2) 11 s4.predecesseurs = [(s1,1),(s2,2)]
12 s5.predecesseurs = [(s1,2),(s3,3),(s4,6)]

3) s2.successeurs[1][1] permet d'accéder au deuxième élément du tuple s2.successeurs[1] qui est (s3,5) donc renvoie la valeur 5.

4) Le site 1 a 4 liens à partir de site2 et 2 liens à partir de site4 donc la valuer de sa popularité est 4+2 = 6.

5)
```
def calculPopularite(self) :
    for i in range(len(self.predecesseurs)) :
        self.popularite += self.predecesseurs[i][1]
    return self.popularite
```
6) On empile les éléments et on retire toujours l'élément le plus ancien (premier arrivé) donc c'est une structure de FILE

7) Il s'agit d'un parcours en largeur.

8) parcoursGraphe(s1) renvoie [s1,s3,s4,s5]

9) 6 maxpopularite = site.popularite
7 siteLePlusPopulaire = site.pop

10) lePlusPopulaire(parcoursGraphe(s1)). Nom renvoie s3.

11) En général, le code proposé est bien adapté pour traiter quelques milliers de sites. La complexité de pour le BFS et O(n) pour la recherche de la popularité sont acceptables pour cette échelle. Tant que ces méthodes restent optimisées et les graphes ne sont pas excessivement denses en termes de nombre d'arêtes, le code devrait fonctionner efficacement pour un grand nombre de sites.

Exercice 2 :

Partie A :

1) Un SGBD permet d'inscrire, de retrouver, de modifier, de trier, de transformer ou d'imprimer les informations de la base de données.

2) Bureau No1 — B — E — A — Prestataire (RIP)

3) Bureau No2 — C — I — H — F — D — A — Prestataire cout : 2,03
Bureau No2 — C — I — G — F — D — A — Prestataire cout : 2,03

Partie B :

4) id_client est un entier qui permet d'identifier de façon unique les enregistrements dans la relation client. Il assure qu'il n'y a pas de doublons pour jouer le rôle de clé primaire.

5) Une clé étrangère permet de mettre en relation deux tables au sein d'une BDD relationnelle. D'assurer l'intégrité référentielle des données. Autrement dit, seules les valeurs devant apparaître dans la base de **données** sont permises.

— reservations a id_client en clé étrangère et référence id_client de la relation clients
a nom_croisiere en clé étrangère et référence nom de la relation croisieres
— croisieres a escale_1, escale_2,escale_3 et escale_4 en clés étrangères et référencent nom de la relation ville

6) Dans la relation reservations #nom_croisiere est INT, il faut le changer en TEXT.

Partie C :

7) Le gestionnaire recherche d'abord l'identifiant de Jean Barc qui doit être 1243 , puis recherche les réservations effectuées par Jean Barc pour vérifier sa demande.

8) SELECT id_reservation
FROM reservations

```
   JOIN clients ON clients.id_client = reservations.id_client
   WHERE nom='BARC' and prenom='JEAN'
```

9)
```
UPDATE reservations
   SET nom_croisiere = ' Croisière Puerto'
  WHERE id_client = 20456
```

10)
```
SELECT nom, prenom, date_naissance
  FROM clients
  JOIN reservations ON reservations.id_client = clients.id_client
  JOIN croisieres ON  croisieres.nom = reservations.nom
  WHERE croisieres.nom = ' Croisière Puerto' OR croisieres.nom = ' Croisière Piano'
```

Exercice 3 :

Partie A :

1) chien40 = Chien(40, 'Duke ', 'Wheel dog', 10)

2)
```
def changer_role(self, nouveau_role) :
        self.role = nouveau_role
```

3) chien40.changer_role('leader')

Partie B :

4)
```
    def retirer_chien(self, numero):
        for chien in self.liste_chiens:
            if chien.id_chien == numero:
                self.liste_chiens.remove(chien)
                    break
```

5) eq11.retirer_chien(46)

6) convert('4h36') renvoie 4 + 36/60 = 4,6

7)
```
def temps_course(equipe) :
        cumul = 0
        for i in range(len(equipe.liste_temps)) :
            cumul += convert( equipe.liste_temps[i])
        return cumul
```

8)

```
                eq1 (276)
               /         \
        eq2 (237)       eq4 (349)
        /       \       /      \
eq3 (189) eq9 (271) eq5 (285) eq8 (352)
      \         /         \
  eq6 (206) eq11 (266) eq7 (291)
           \
         eq10 (224)
```

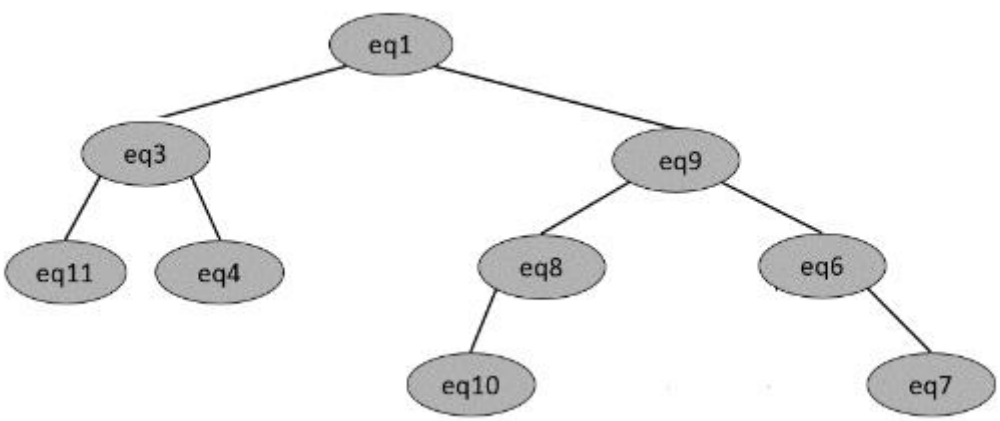
9) le parcours suffixe ou postfixe

10) La fonction inserer est récursive car elle s'appelle elle-même.

11) 6 arb.gauche = Noeud(eq)
8 inserer(arb.gauche, eq)
13 inserer(arb.droit, eq)

12) 3 return arbre.gauche.equipe
5 return est_gagnante(arbre.gauche)

13)



14) def rechercher(arbre, equipe):
if arbre is None:
return False

if equipe == arbre.equipe:
return True

```
if temps_course(equipe) < temps_course(arbre.equipe):
        return rechercher(arbre.gauche, equipe)
else:
        return rechercher(arbre.droite, equipe)
```