Correction CNC 2018 - TSI

Sommaire

Partie I :

Q-1 :
clé primaire de la table Membres : **id**
clé primaire de la table Liens **: id_enfant** ou **(id_parent, id_enfant)**

Q-2 :
```
Select prenom, genre
From Membre
Where generation=0
```

Q-3 :
```
Select distinct prenom, generation
From Membres
Where genre='M' and prenom like 'A%E%D'
```

Q-4 :
```
Select max(generation)+1
From Membres
```

Q-5 :
```
Select prenom, genre
From Membres, Liens
Where id=id_fils and id_parent=15
Order by prenom
```

Q-6 :
```
Select id, nom, generation, count(*)
From Membres , Liens
Where id = id_parent
Group by id
Having count(*) >= 3
Order by Count(*) desc
```

Q-7 :
```
Create Table Membres_1
   ( id Entier, prenom Texte, genre Texte, generation Entier)
```

Q-8 :
```
Insert into Membres_1
     Select *
     From Membres
     Where prenom = (Select prenom
                          From Membres
                          Where generation=0 )
                    And generation !=0
```

Partie II :

Q1 : 2 bits sont suffisants pour coder 4 éléments

Q2 :
Exemple de codes:
00 pour 'A', 01 pour 'C', 10 pour 'G', 11 pour 'T'

Q3 :
```
def prefixe(M,S):
    m=len(M)
    return  M==S[:m]
```

Q4 : m=len(M)
pire cas : M=S ou (M[:m-1]=S[:m-1] et M[-1]!=S[m-1])
La fonction effectue m comparaisons élémentaires
Complexité linéaire O(m)

Q5 :
```
def suffixes(S):
    return [(i,S[i:]) for i in range(len(S))]
```

Q6 :
```
def tri_liste(L):
    n=len(L)
    if n<=1: return L
    A,B=[],[]
    for i in range(1,n):
        if L[i]<L[0] : A.append(L[i])
        else: B.append(L[i])
    return tri_liste(A)+[L[0]]+tri_liste(B)
```

Q7 :
```
def recherche_dichotomique(M,L):
    n=len(L)
    if n==0: return None
    m=n//2
    S=L[m]
    if prefixe(M,S[0]): return S[1]
    elif M<S[0]: return recherche_dicho(M,L[:m])
    else: return recherche_dicho(M,L[m+1:])
```

Q8 : m,k=len(M),len(L)
recherche_dichotomique : Complexité quasi-linéaire : O(m*log(k))
Justification :

Q9 :

```
def recherche_dichotomique(M , L):
        d,f = 0,len(L)-1
         while d<=f :
                m= (d+f)//2
                if prefixe(M , L[m][0]): return L[m][1]
                elif M< L[m][0]: f=m-1
                else: d=m+1
        return None
```

Q10 :

m ,k = len(M),len(L)
recherche_dichotomique : Complexité quasi-linéaire : O(m*log(k))
Exemple : S='AAAAAAAAAAA' et M='AAAAAT'

Partie III :

Q-1 :

$x_{n+1} = x_n – 2*h\ f(x_n)/(f(x_n+h) – f(x_n-h))$

Q-2 :

```
def newton(f , x0 , eps , h):
        x=x0 – 2*h*f(x0)/(f(x0+h) – f(x0–h))
        while abs(x – x0)> eps:
                x0=x
                x=x0 – 2*h*f(x0)/(f(x0+h) – f(x0–h))
         return x
```

Q-3 :

```
def g(x):
    return x**n – a
```

Q-4 :

```
X = linspace(0 , 2 , 500)
Y = g(X)
plot (X , Y)
```

Q-5 :

```
x = newton(g , a ,10**-12 , 10**-5)
print(x)
```