
Numérique et Science Informatique

Bac Blanc - Corrigé

20 février 2023

Exercice 1 : Base de données

8 points

1. La table **Articles** utilise des clés étrangères des tables **Auteurs** et **Themes**. Si ces dernières sont vides, il n'est pas possible de lier les valeurs et donc on ne peut pas insérer de valeurs.
2. On saisit **INSERT INTO** Traitements (article, theme) **VALUES** (2, 4)
3. On saisit **UPDATE** Auteurs **SET** nom = "Jèraus" **WHERE** idAuteur = 2
4. a. Le titre des articles parus après le 1^{er} janvier 2022 inclus :

```
1 | SELECT titre
2 | FROM Articles
3 | WHERE dateParution >= 20220101
```

- b. Le titre des articles écrits par l'auteur Étienne Zola :

```
1 | SELECT titre
2 | FROM Articles
3 | WHERE auteur = 3
```

- c. Le nombre d'articles écrits par l'auteur Jacques Pulitzer (présent dans la table **Auteurs** mais on ne connaît pas son **idAuteur**) :

```
1 | SELECT count(*)
2 | FROM Articles
3 | JOIN auteurs ON Articles.auteur = Auteurs.idAuteur
4 | WHERE nom LIKE "Pulitzer" AND prenom LIKE "Jacques"
```

- d. Les dates de parution des articles traitant du thème « Sport »

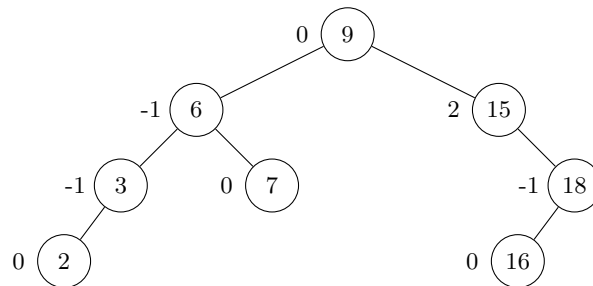
```
1 | SELECT date
2 | FROM Articles
3 | JOIN Traitements ON Articles.idArticle = Traitements.article
4 | JOIN Themes ON Traitements.theme = Themes.idTheme
5 | WHERE Themes.themes LIKE "Sport"
```

Exercice 2 : Arbres binaires équilibrés

12 points

Partie A :

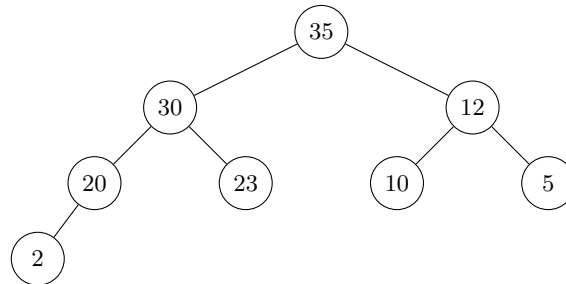
1. a. On obtient :



b. Cet arbre n'est pas équilibré car le nœud de valeur 15 a une balance de 2.

2. a. On obtient [0, 45, 40, 48, 17, 43, 46, 49, 14, 19]

b. On obtient :



3. a. `f(arbre, 1)` renvoie 3. En effet, si l'arbre est vide ou si la valeur de sa racine est **None**, on renvoie 0. Dans le cas contraire, on renvoie 1 plus de maximum des résultats des sous-arbres gauches et droits (indices $2*i$ et $2*i+1$). On calcule ainsi la hauteur de l'arbre.

b. La fonction `f` permet de calculer la hauteur d'un arbre.

4.

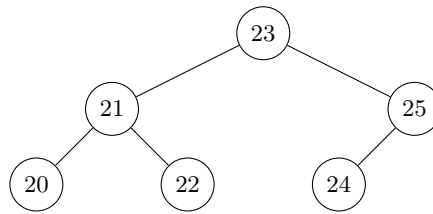
```

1 def estEquilibre(arbre: list, i : int) -> bool:
2     if i >= len(arbre) or arbre[i] is None:
3         return True
4     else:
5         balance = f(arbre, 2*i+1) - f(arbre, 2*i)
6         reponse = balance in [-1, 0, 1]
7         return reponse and estEquilibre(arbre, 2*i) and estEquilibre(arbre, 2*i+1)
  
```

Partie B :

1.
 - Parcours *préfixe* : 45, 40, 17, 14, 19, 43, 48, 46, 49
 - Parcours *infixe* : 14, 17, 19, 40, 43, 45, 46, 48, 49
 - Parcours *suffixe* : 14, 19, 17, 43, 40, 46, 49, 48, 45

2. On obtient :



3. On propose :

```

1 def infixe(arbre: list) -> list:
2     pile = []
3     visites = []
4     n = 1
5     repetition = True
6     while repetition :
7         while n < len(arbre) and arbre[n] is not None :
8             pile.append(n)
9             n = 2*n
10        if len(pile) == 0 :
11            repetition = False
12        else :
13            n = pile.pop()
14            visites.append(arbre[n])
15            n = 2*n+1
16    return visites
  
```

4. On propose :

```

1 def construitABR(i, ordre):
2     while len(nouveau) <= i:
3         nouveau.append(None)
4
5     i_milieu = len(ordre)//2
6     nouveau[i] = ordre[i_milieu]
7
8     gauche = ordre[:i_milieu]
9     if len(gauche) > 0:
10        construitABR(2*i, gauche)
11
12    droite = ordre[(i_milieu+1):]
13    if len(droite) > 0:
14        construitABR(2*i+1, droite)
  
```