Epreuve écrite type BAC

20 février 2023

Année scolaire 2022-2023

DS7 Exercice 1 : SQL

Question 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.

DS7 Exercice 1 : SQL

Question 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.

Question 2

On saisit INSERT INTO Traitements (article, theme) VALUES (2, 4)

DS7 Exercice 1 : SQL

Question 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.

Question 2

On saisit INSERT INTO Traitements (article, theme) VALUES (2, 4)

Question 3

On saisit UPDATE Auteurs SET nom="Jèraus" WHERE idAuteur = 2

DS7 Exercice 1 : SQL

Question 4.a

Le titre des articles parus après le 1^{er} janvier 2022 inclus :

```
SELECT titre
FROM Articles
WHERE dateParution >= 20220101
```

DS7 Exercice 1 : SQL

Question 4.a

Le titre des articles parus après le 1^{er} janvier 2022 inclus :

```
SELECT titre
FROM Articles
WHERE dateParution >= 20220101
```

Question 4.b

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

```
SELECT titre
FROM Articles
WHERE auteur = 3
```

DS7 Exercice 1 : SQL

Question 4.a

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

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

DS7 Exercice 1 : SQL

Question 4.a

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

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

Question 4.b

Les dates de parution des articles traitant du thème « Sport » :

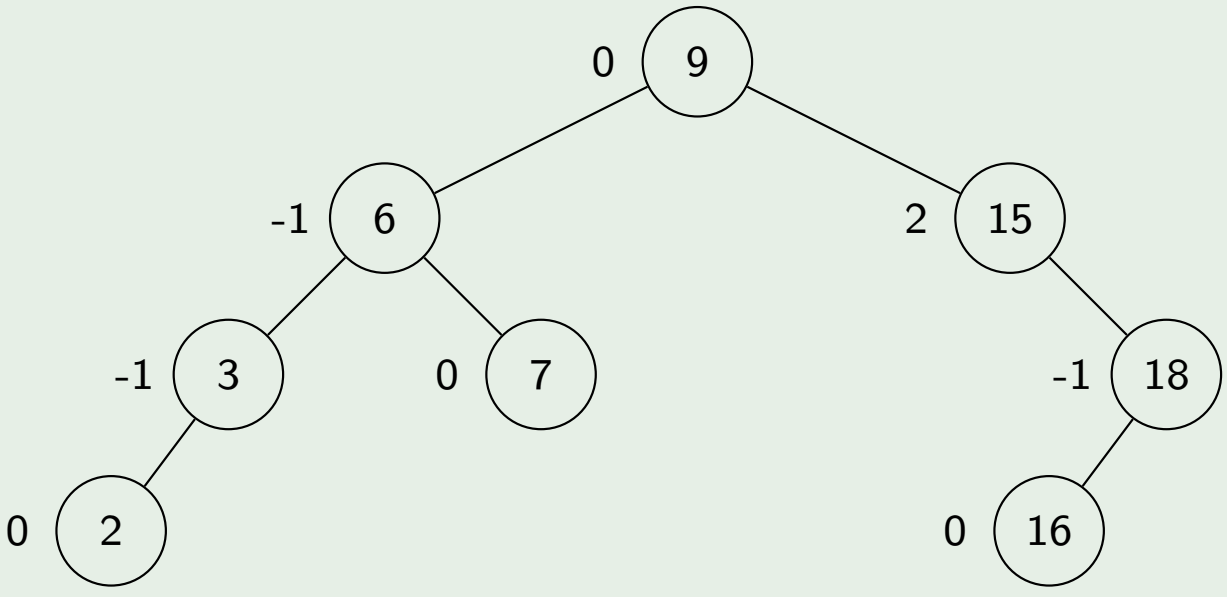
```
SELECT dateParution
FROM Articles
JOIN Traitements ON Articles.idArticle = Traitements.article
JOIN Themes ON Traitements.theme = Themes.idTheme
WHERE Themes.themes LIKE "Sport"
```

.

DS7 Exercice 2 : Arbres binaires équilibrés

Question A.1.a

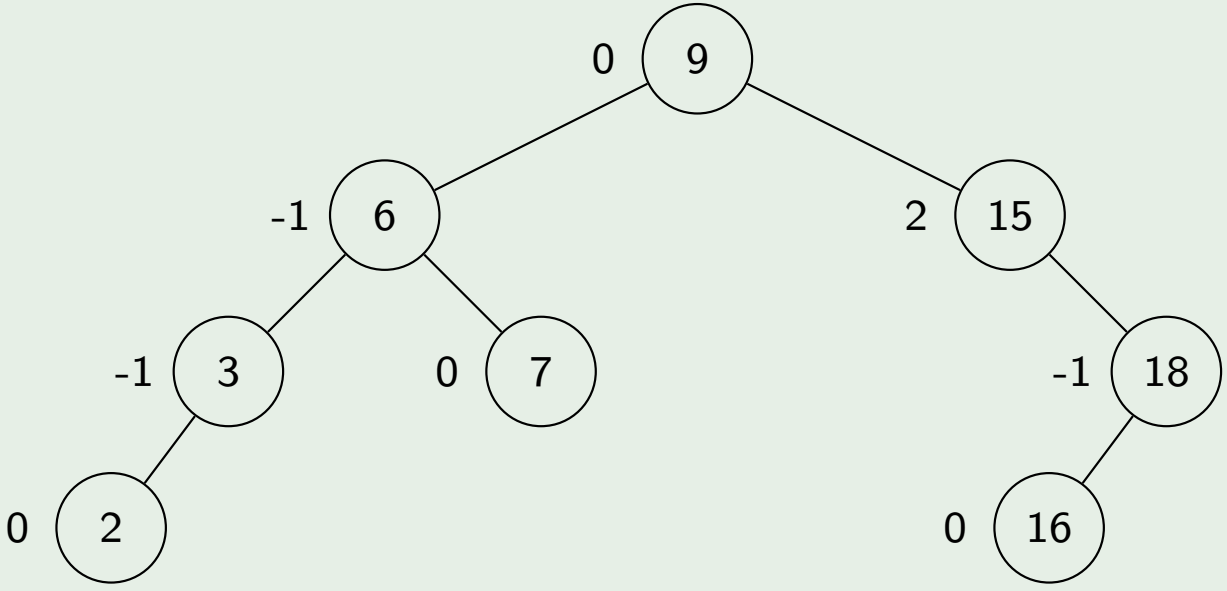
On obtient :



DS7 Exercice 2 : Arbres binaires équilibrés

Question A.1.a

On obtient :



Question A.1.b

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

DS7 Exercice 2 : Arbres binaires équilibrés

Question A.2.a

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

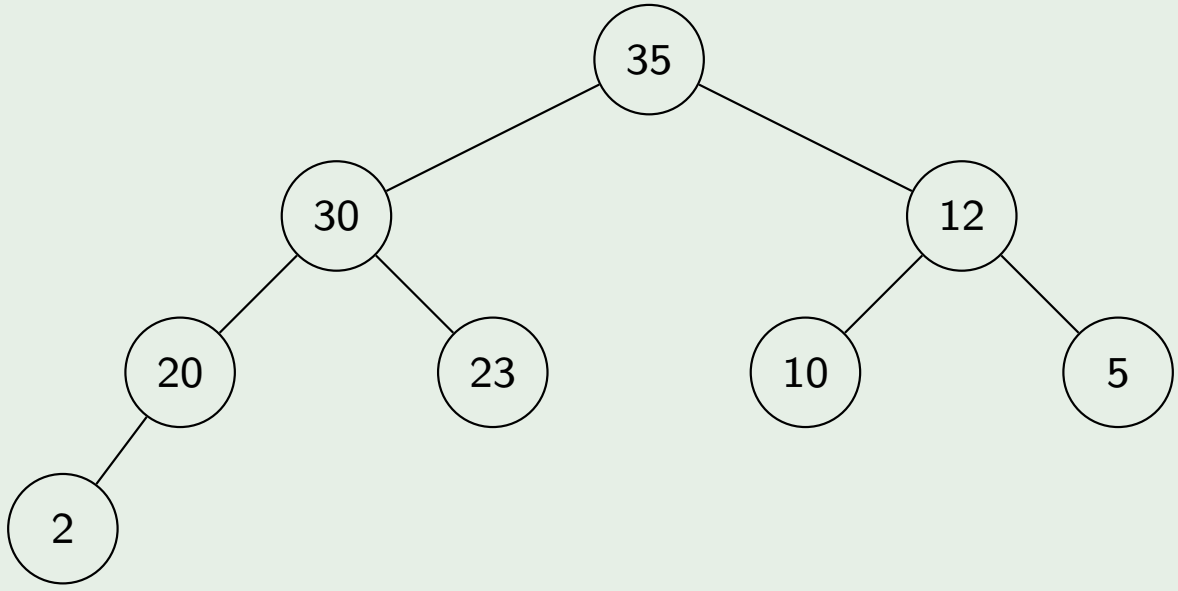
DS7 Exercice 2 : Arbres binaires équilibrés

Question A.2.a

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

Question A.2.b

On obtient :



DS7 Exercice 2 : Arbres binaires équilibrés

Question A.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.

DS7 Exercice 2 : Arbres binaires équilibrés

Question A.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.

Question A.3.b

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

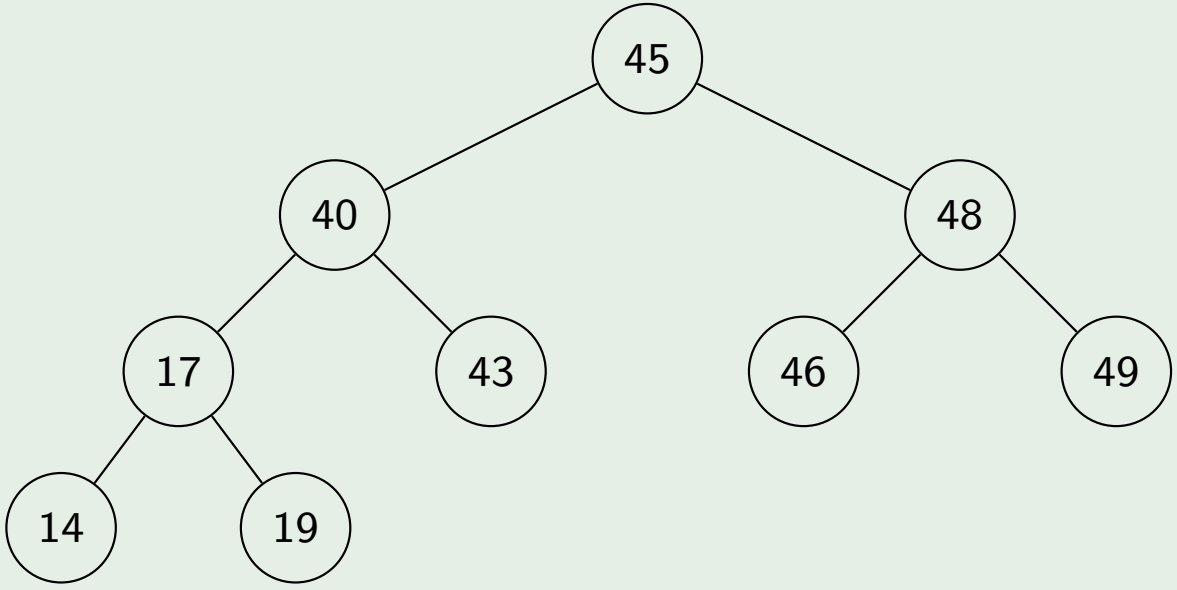
DS7 Exercice 2 : Arbres binaires équilibrés

Question A.4

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

DS7 Exercice 2 : Arbres binaires équilibrés

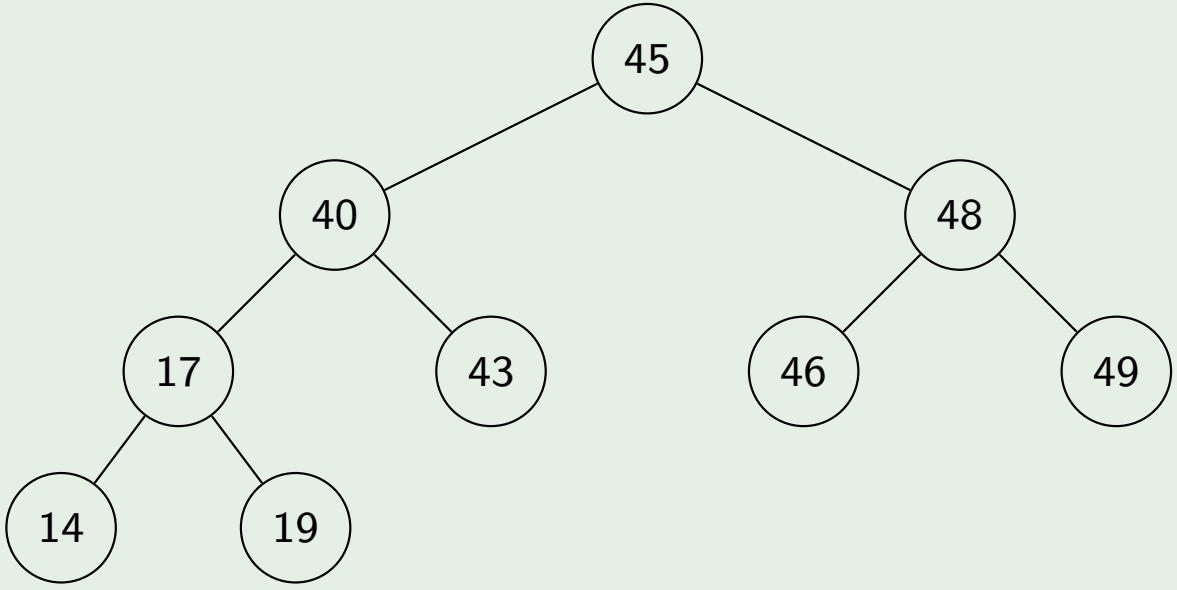
Question B.1



- Parcours *préfixe* : 45, 40, 17, 14, 19, 43, 48, 46, 49

DS7 Exercice 2 : Arbres binaires équilibrés

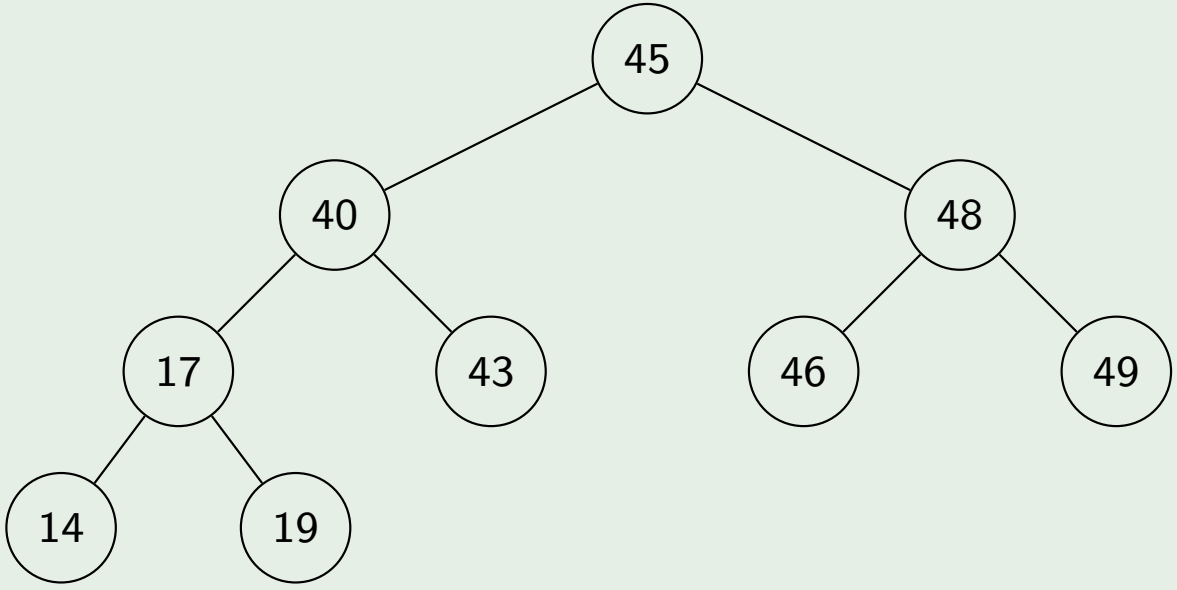
Question B.1



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

DS7 Exercice 2 : Arbres binaires équilibrés

Question B.1

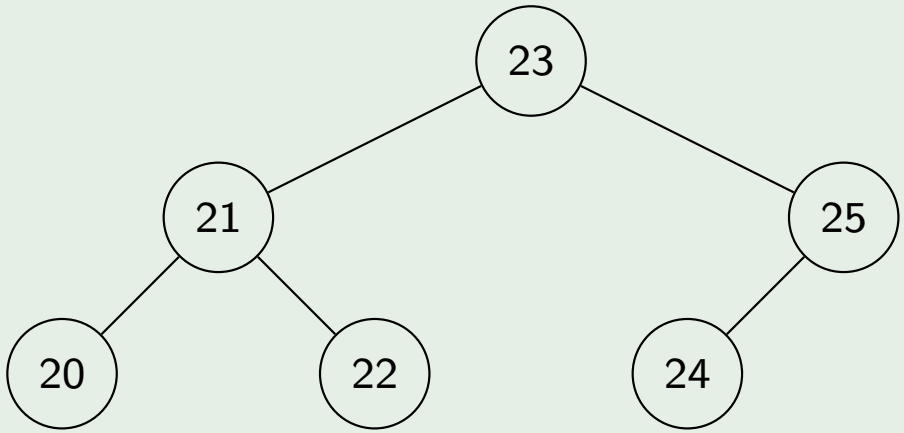


- Parcours *préfixe* : 45, 40, 17, 14, 19, 43, 48, 46, 49
- Parcours *infixe* : 14, 17, 19, 40, 43, 45, 46, 48, 49
- Parcours *suffixe* : 14, 19, 17, 43, 40, 46, 49, 48, 45

DS7 Exercice 2 : Arbres binaires équilibrés

Question B.2

On obtient :



DS7 Exercice 2 : Arbres binaires équilibrés

Question B.3

```
def infixe(arbre: list) -> list:
    pile = []
    visites = []
    n = 1
    repetition = True
    while repetition :
        while n < len(arbre) and arbre[n] is not
            None :
            pile.append(n)
            n = 2*n
        if len(pile) == 0 :
            repetition = False
        else :
            n = pile.pop()
            visites.append(arbre[n])
            n = 2*n+1
    return visites
```

DS7 Exercice 2 : Arbres binaires équilibrés

Question B.4

```
1def construitABR(i, ordre):
2#Ajoute la valeur None à la liste nouveau jusqu'à
      ce qu'elle soit de longueur i+1
3     while len(nouveau) <= i:
4          nouveau.append(None)
5# Donne la valeur du milieu de ordre à nouveau[i]
6     i_milieu = len(ordre)//2
7     nouveau[i] = ordre[i_milieu]
8# Détermine la moitié gauche de ordre
9     gauche = ordre[:i_milieu]
10# Si celle-ci est non-vide, appelle construireABR
     (2*i, moitié gauche de ordre)
11    if len(gauche) > 0:
12         construitABR(2*i, gauche)
13    droite = ordre[(i_milieu+1):]
14    if len(droite) > 0:
15         construitABR(2*i+1, droite)
```