

[← Index des sujets 2021](#)

21-NSIJ1ME1 : Corrigé

Année : **2021**

Centre : **Métropole**

Jour : **1**

Enoncé : [!\[\]\(e3f8612927870f2e0f9f5989e6dd3064_img.jpg\) PDF](#)

1. Exercice 1

arbres binaires de recherche

1. a. Il y a 4 feuilles, d'étiquette 12, val , 21 et 32.
- b. Le sous-arbre gauche du nœud 23 est 19-21.
- c. La hauteur de l'arbre est 4. Sa taille est 9.
- d. Les valeurs possibles de val sont 16 et 17.
2. a. Parcours infixé : 12-13-15-16-18-19-21-23-32
- b. Parcours suffixe : 12-13-16-15-21-19-32-23-18
3. a.

```
graph TD
    A(18) --> B(12)
    B --> D(" ")
    B --> E(13)
    A --> C(19)
    C --> F(" ")
    C --> G(21)
    D --> H(" ")
    D --> D1(" ")
    E --> E1(" ")
    E --> E2(15)
    F --> F1(" ")
    F --> M(" ")
    M --> G1(" ")
    M --> G2(" ")
    E2 --> I(" ")
    E2 --> J(16)
    G --> K(" ")
    G --> L(32)
    L --> N(23)
    L --> O(" ")
    linkStyle 1 stroke-width:0px;
    style D opacity:0;
    linkStyle 4 stroke-width:0px;
    style F opacity:0;
```

```

linkStyle 6 stroke-width:0px;
style H opacity:0;
linkStyle 7 stroke-width:0px;
style D1 opacity:0;
linkStyle 8 stroke-width:0px;
style E1 opacity:0;
linkStyle 10 stroke-width:0px;
style F1 opacity:0;
linkStyle 11 stroke-width:0px;
style M opacity:0;
linkStyle 12 stroke-width:0px;
style G1 opacity:0;
linkStyle 13 stroke-width:0px;
style G2 opacity:0;
linkStyle 14 stroke-width:0px;
style I opacity:0;
linkStyle 16 stroke-width:0px;
style K opacity:0;
linkStyle 19 stroke-width:0px;
style O opacity:0;

```

b.

Script Python

```

racine = Noeud(18)
racine.insere([15, 13, 12, 16, 23, 32, 19, 21])

```

(d'autres solutions sont possibles)

c. Bloc 3 - Bloc 2 - Bloc 1

4.

Script Python

```

1  class Noeud():
2      def __init__(self, v):
3          self.ag = None
4          self.ad = None
5          self.v = v
6
7      def insere(self, v):
8          n = self
9          est_insere = False
10         while not est_insere:
11             if v == n.v:
12                 est_insere = True
13             elif v < n.v:
14                 if n.ag != None:
15                     n = n.ag
16                 else:
17                     n.ag = Noeud(v)
18                     est_insere = True
19             else:
20                 if n.ad != None:
21                     n = n.ad
22                 else:
23                     n.ad = Noeud(v)
24                     est_insere = True
25

```

```

26     def insere_tout(self, vals):
27         for v in vals:
28             self.insere(v)
29
30     def recherche(self, v):
31         arbre = self
32         while not arbre is None:
33             if arbre.v == v:
34                 return True
35             if v < arbre.v:
36                 arbre = arbre.ag
37             else:
38                 arbre = arbre.ad
39         return False
40
41
42     # version récursive (non demandée)
43
44     def recherche_rec(self, v):
45         if self is None:
46             return False
47         if self.v == v:
48             return True
49         if v < self.v:
50             if self.ag is not None:
51                 return self.ag.recherche_rec(v)
52             else:
53                 return False
54         else:
55             if self.ad is not None:
56                 return self.ad.recherche_rec(v)
57             else:
58                 return False
59
60
61 racine = Noeud(18)
62 racine.insere_tout([12, 13, 15, 14, 19, 21, 32, 23])
63 print(racine.recherche(149))
64 print(racine.recherche(12))

```

2. Exercice 2

gestion des processus, opérateurs booléens

1. Réponse b
2. Réponse c
3. Réponse b
4. Réponse d

Partie B

P3	P3	P2	P1	P1	P1	P2	P2	P3	P3
0	1	2	3	4	5	6	7	8	9

1. Il s'agit du scénario 2 car nous nous retrouvons dans la situation où P1 possède R1 et attend R2 avant de pouvoir continuer et P3 possède R2 et attend R1 avant de pouvoir continuer.

2. Il s'agit du scénario 2 car nous nous retrouvons dans la situation où P1 possède R1 et attend R2 avant de pouvoir continuer et P3 possède R2 et attend R1 avant de pouvoir continuer.

Partie C

1. a. 0100 0110 => 46 en hexa => caractère F

0110 0011 => 63 en hexa => caractère c

donc cF

b. 0b 1000 1101 1011 0110

2. a.

a	b	(a XOR b) XOR b
0	0	0
0	1	0
1	0	1
1	1	1

- b. On peut remarquer que (a xor b) xor b permet de retrouver a, donc si a correspond au message non chiffré et a xor b correspond au message chiffré, un (a xor b) xor b permet donc de retrouver le message non chiffré. Si on appelle m le message non chiffré, m' le message chiffré et k la clé de chiffrement, un m' xor k permettra de retrouver m.

3. Exercice 3

base de données et langage SQL

1. Systèmes de gestion de base de données (SGBD).

2. a. En supprimant la ligne du train 1241 de la table Train, on ferait perdre la référence de la clé étrangère de la table Reservation. Cette opération est donc interdite.

- b. On ne peut pas enregistrer une ligne dans la table Reservation si la valeur de l'attribut numT n'apparaît pas dans la table Train pour le champ numT.

3. a.

 Requête SQL

```
SELECT numT
FROM Train
WHERE destination = "Lyon";
```

b.

Requête SQL

```
INSERT INTO Reservation VALUES (1307, "Turing", "Alan", 33, 654);
```

c.

Requête SQL

```
UPDATE Train
SET horaireArrivee = "08:11"
WHERE numT = 7869;
```

4. Elle permet de connaître le nombre de réservations de Grace Hopper.

5. Requête SQL

```
SELECT destination, prix
FROM Train
JOIN Reservation ON Train.numT = Reservation.numT
WHERE nomClient = "Hopper" AND prenomClient = "Grace";
```

4. Exercice 4

algorithme de tri fusion et méthode diviser pour régner

1. a. $O(n \log(n))$
- b. Tri par insertion en $O(n^2)$. Le coût du tri fusion est plus intéressant.

2. Script Python

```
[7, 4, 2, 1, 8, 5, 6, 3]
[7, 4, 2, 1]
[7, 4]
[2, 1]
[8, 5, 6, 3]
[8, 5]
[6, 3]
```

3. Script Python

```
def moitie_droite(L):
    md = []
    n = len(L)
    mil = n // 2
```

```
for i in range(mil, n):
    md.append(L[i])
return md
```

4.  Script Python

```
if e1 <= e2:
    L.append(e1)
    i1 = i1 + 1
else:
    L.append(e2)
i2 = i2 + 1
```

5. Exercice 5

réseaux et protocoles de routage

1. a. Il l'envoie vers R2 car la passerelle de la table de routage correspond à une adresse du sous-réseau entre R1 et R2.
b. R1-R2-R6
2. a. R1-R3-R4-R6
b. La ligne de R1.
3. a. $C = \frac{10^9}{10^7} = 100$
b. R1-R2-R4-R5-R6
c. Les lignes de R2 et R4.