Bac Général Maths + NSI (Mathématiques et Numérique et sciences informatiques)

CORRIGÉ NON OFFICIEL

Proposition de corrigé rédigée par FormaV avec l'aide d'outils d'intelligence artificielle. Elle peut contenir des erreurs et ne remplace ni le corrigé ni le barème officiels de l'examen.

Les sujets officiels des sessions passées et nos sujets d'entraînement sont classés sur https://bac-maths-nsi.fr/annales-sujets-et-corriges

Document mis en ligne par FormaV®. Reproduction ou diffusion sans autorisation interdite.

Numérique et sciences informatiques

Session 2023 • Amérique du Nord • Jour 1

Code sujet **23-NSIJ1AN1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Traitement de données en table et bases de données
13 réponses • sujet p. 2–6

Exercice 2 — Processus, ordonnancement et logique booléenne
9 réponses • sujet p. 7–9

Exercice 3 — Programmation orientée objet et arbres binaires de recherche
10 réponses • sujet p. 10–13

Les numéros, parties et pages renvoient au sujet original. Chaque réponse rappelle la consigne, explique la démarche et donne le résultat.

FormaV - document d'entraînement non officiel

Exercice 1

5 points

Traitement de données en table et bases de données

A

Question 1.a)

Sujet p. 2, 3

> Accès aux informations du tableau : donner, sans justifier, la valeur à laquelle on accède avec l'instruction tab_lieux[0]["insee"].

`tab_lieux[0]` désigne le premier dictionnaire du tableau. Dans celui-ci, la valeur associée à la clé `insee` est la chaîne « 1024 ».

> **RÉPONSE À RETENIR**
>
> `"1024"`

Question 1.b)

Sujet p. 2, 3

> Écrire l'instruction qui permet d'obtenir la valeur « La Boisse ».

« La Boisse » figure dans le deuxième dictionnaire, d'indice 1. On accède à son adresse avec la clé `ad_lieu`.

> **RÉPONSE À RETENIR**
>
> `tab_lieux[1]["ad_lieu"]`

Question 2. Proposition 1

Sujet p. 2, 3

> Trois blocs d'instructions sont proposés pour parcourir le tableau et afficher le nombre de places des lieux de covoiturage. Deux seulement sont corrects. Indiquer, sans justifier, les deux propositions correctes. Proposition 1 : for i in range(len(tab_lieux)): print(dico["nb_places"]).

La boucle définit `i`, mais utilise ensuite `dico`, qui n'est pas défini dans ce bloc. Elle ne permet donc pas d'afficher correctement les nombres de places.

> **RÉPONSE À RETENIR**
>
> La proposition 1 n'est pas correcte.

Question 2. Proposition 2

Sujet p. 2, 3

> Trois blocs d'instructions sont proposés pour parcourir le tableau et afficher le nombre de places des lieux de covoiturage. Deux seulement sont corrects. Indiquer, sans justifier, les deux propositions correctes. Proposition 2 : for i in range(len(tab_lieux)): print(tab_lieux[i]["nb_places"]).

La boucle parcourt les indices du tableau. À chaque tour, `tab_lieux[i]` désigne un dictionnaire, et `["nb_places"]` permet d'accéder au nombre de places correspondant.

RÉPONSE À RETENIR

La proposition 2 est correcte.

Question 2. Proposition 3

Sujet p. 2, 3

> Trois blocs d'instructions sont proposés pour parcourir le tableau et afficher le nombre de places des lieux de covoiturage. Deux seulement sont corrects. Indiquer, sans justifier, les deux propositions correctes. Proposition 3 : for dico in tab_lieux: print(dico["nb_places"]).

La boucle parcourt directement les dictionnaires contenus dans `tab_lieux`. À chaque tour, `dico` désigne l'un de ces dictionnaires et permet d'accéder à son nombre de places.

RÉPONSE À RETENIR

La proposition 3 est correcte.

Question 3.

Sujet p. 2, 3

> Décrire, dans le contexte de l'exercice, ce que renvoie la fonction fournie lors de l'appel fonction_inconnue(tab_lieux, 100, "Sortie autoroute").

La fonction initialise `k` à 0 et l'incrémente de 1 pour chaque lieu qui vérifie simultanément les conditions `nb_places >= n` et `type == nom_type`. Pour l'appel donné, elle compte donc les lieux ayant au moins 100 places et de type « Sortie autoroute ». Parmi les trois lieux illustrés, aucun ne vérifie ces deux conditions : le lieu de sortie d'autoroute a 15 places. Toutefois, le tableau est suivi de points de suspension et ses autres éléments ne sont pas fournis ; le résultat numérique sur le tableau complet ne peut donc pas être déterminé.

RÉPONSE À RETENIR

La fonction renvoie le nombre de lieux de type « Sortie autoroute » ayant au moins 100 places. La valeur numérique ne peut pas être déterminée à partir des seuls éléments illustrés.

Question 4.

Sujet p. 2, 3, 4

> La fonction insee_max_places permet de renvoyer le code INSEE du lieu de covoiturage possédant le plus grand nombre de places. Recopier et compléter le code proposé.

On prend le premier lieu comme maximum provisoire, puis on parcourt le tableau. Lorsqu'un lieu possède strictement plus de places que le maximum courant, on met à jour le maximum et le code INSEE associé. Cette solution suppose que le tableau n'est pas vide, condition nécessaire pour initialiser les variables à partir de son premier élément. En cas d'égalité au maximum, elle renvoie le code INSEE d'un des lieux ayant ce maximum.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

```
def insee_max_places(table):
    """
    Entrée : un tableau de dictionnaires
    Sortie : une chaîne de caractères (le code INSEE du lieu
             possédant le plus grand nombre de places)
    """
    maxi = table[0]["nb_places"]
    code_insee = table[0]["insee"]
    for dico in table:
        if dico["nb_places"] > maxi:
            maxi = dico["nb_places"]
            code_insee = dico["insee"]
    return code_insee
```

Question 5.

Sujet p. 2, 4

La fonction moyenne_par_type(table, nom_type) prend en paramètres une table de lieux de covoiturage au format tableau de dictionnaires et le nom d'un type de lieu. Elle renvoie, parmi les lieux dont le type est nom_type, la moyenne du nombre de places. Recopier et compléter le code ; les points de suspension peuvent correspondre à une ou plusieurs lignes de programme. L'exemple fourni comporte trois lieux : deux de type « Supermarché », ayant 10 et 20 places, et un de type « Parking municipal », ayant 100 places ; moyenne_par_type(tabl, "Supermarché") vaut 15.

On additionne les nombres de places des lieux correspondant au type demandé, tout en comptant ces lieux. On divise ensuite la somme par cet effectif. La spécification ne précise pas le résultat s'il n'existe aucun lieu de ce type ; le code ci-dessous suppose donc que l'effectif est non nul.

Dans l'exemple, les deux lieux « Supermarché » ont 10 et 20 places : la moyenne est $(10 + 20)/2 = 15$.

RÉPONSE À RETENIR

```
def moyenne_par_type(table, nom_type):
    """
    Entrée : un tableau de dictionnaires et une chaîne de caractères
             (type de lieu)
    Sortie : la moyenne du nombre de places pour le type choisi
    """
    total = 0
    effectif = 0
    for dico in table:
        if dico["type"] == nom_type:
            total = total + dico["nb_places"]
            effectif = effectif + 1
    moyenne = total / effectif
    return moyenne
```

B

Question 6.

Sujet p. 5, 6

> Dans la relation COMMUNE, expliquer pourquoi deux communes ne peuvent pas posséder le même code INSEE code_insee.

Dans le schéma relationnel, `code_insee` est la clé primaire de la relation `COMMUNE`. Une clé primaire identifie chaque enregistrement de manière unique. Deux communes ne peuvent donc pas avoir le même code INSEE.

> **RÉPONSE À RETENIR**
>
> Le code INSEE est la clé primaire de `COMMUNE` et est donc unique pour chaque commune.

Question 7.

Sujet p. 5, 6

> Écrire une requête SQL permettant d'obtenir l'identifiant id_lieu et le code INSEE code_insee des lieux de covoiturage de type « Sortie autoroute ».

On sélectionne les deux colonnes demandées dans `LIEU` et on conserve uniquement les lignes dont le type est « Sortie autoroute ».

```
SELECT id_lieu, code_insee
FROM LIEU
WHERE type = 'Sortie autoroute';
```

Dans l'extrait de `LIEU`, les deux types affichés sont « Supermarché » et « Parking municipal » : aucune ligne ne satisfait la condition, donc le résultat sur cet extrait est vide.

> **RÉPONSE À RETENIR**
>
> La requête renvoie les identifiants et codes INSEE des lieux de type « Sortie autoroute ».

Question 8.

Sujet p. 5, 6

> Écrire une requête SQL permettant de compter le nombre de lieux de covoiturage se trouvant dans le département « JURA ».

Le département figure dans `COMMUNE`. Il faut donc joindre chaque lieu à sa commune avec le code INSEE, filtrer les communes du département « JURA », puis compter les lieux correspondants.

```
SELECT COUNT(LIEU.id_lieu)
FROM LIEU
JOIN COMMUNE ON LIEU.code_insee = COMMUNE.code_insee
WHERE COMMUNE.departement = 'JURA';
```

Sur les seules lignes affichées dans les extraits, la jointure trouve le lieu de code INSEE « 01001 », rattaché au département « AIN ». Le lieu de code « 01049 » n'a pas de commune correspondante dans l'extrait de `COMMUNE`. La requête appliquée aux extraits affichés donne donc 0. Les extraits étant partiels, ce résultat ne permet pas de connaître le compte dans les relations complètes.

FormaV - document d'entraînement non officiel

COUNT(LIEU.id_lieu)
0

RÉPONSE À RETENIR

La requête compte les lieux rattachés à une commune du département « JURA ». Sur les lignes affichées dans les extraits, le compte est 0 ; le résultat sur les relations complètes n'est pas déterminable avec ces seuls extraits.

Question 9.a)

Sujet p. 5, 6

La commune de code INSEE « 40714 » vient d'être intégrée à la commune voisine « 40146 » ; la base de données doit donc être mise à jour. Une requête DELETE FROM COMMUNE WHERE code_insee = "40714"; a été saisie et le logiciel a renvoyé une erreur. Expliquer pourquoi.

`LIEU.code_insee` est une clé étrangère qui référence la clé primaire `COMMUNE.code_insee`. Si au moins un lieu est encore rattaché au code « 40714 », supprimer la commune laisserait ce lieu sans commune référencée. La contrainte de clé étrangère interdit alors la suppression. Il faut d'abord réaffecter les lieux concernés à « 40146 ».

RÉPONSE À RETENIR

La suppression est refusée si des lignes de `LIEU` référencent encore le code « 40714 », car cela violerait la contrainte de clé étrangère.

Question 9.b)

Sujet p. 5, 6

Les informations liées à la commune de code INSEE « 40714 » devront désormais être rattachées à la commune de code INSEE « 40146 ». Écrire une requête permettant de mettre à jour la table LIEU.

On remplace dans `LIEU` le code INSEE de la commune absorbée par celui de la commune de rattachement.

RÉPONSE À RETENIR

```
UPDATE LIEU
SET code_insee = '40146'
WHERE code_insee = '40714';
```


FormaV - document d'entraînement non officiel

Exercice 2

3 points

Processus, ordonnancement et logique booléenne

A

Question 1.

Sujet p. 7, 8

> En utilisant les données de l'extrait de terminal, déterminer le nom de l'application associée au processus dont l'identifiant est 5468.

On repère la ligne dont le PID vaut 5468, puis on lit la colonne CMD : elle indique `/snap/gimp/393/usr/bin/gimp`. Le nom de l'application est le dernier élément de ce chemin.

> **RÉPONSE À RETENIR**
>
> L'application associée au processus 5468 est **gimp**.

Question 2.

Sujet p. 7, 8

> En utilisant les données de l'extrait de terminal, déterminer l'identifiant du processus qui a sollicité le processeur le plus longtemps.

On compare les valeurs de la colonne TIME. La plus grande durée indiquée dans l'extrait est `00:00:51`, associée au PID 1770. Les autres durées affichées sont au plus égales à `00:00:02`.

> **RÉPONSE À RETENIR**
>
> Le processus qui a sollicité le processeur le plus longtemps a pour identifiant **1770**.

Question 3.

Sujet p. 7, 8

> En utilisant les données de l'extrait de terminal, déterminer l'identifiant du processus qui a le plus d'enfants.

Les enfants d'un processus sont les processus dont le PPID est égal au PID de ce parent. Dans l'extrait, les processus de PID 1752, 1770, 5564 et 5615 ont tous le PPID 1611. On en compte donc quatre. Aucun autre PID de l'extrait n'a autant d'enfants.

> **RÉPONSE À RETENIR**
>
> Le processus qui a le plus d'enfants est celui de PID **1611**, avec quatre enfants dans l'extrait.

Question 4.

Sujet p. 7, 8

> En utilisant les données de l'extrait de terminal, déterminer le nom du programme dont est issue la commande ps.

La dernière ligne de l'extrait correspond au processus de PID 5629 et sa colonne CMD contient `ps -eo user,pid,ppid,time,cmd`. Le nom du programme à l'origine de cette commande est `ps`.

> **RÉPONSE À RETENIR**
>
> La commande est issue du programme **ps**.

FormaV - document d'entraînement non officiel

Question 5.

Sujet p. 7, 8

> Déterminer la succession des identifiants des processus qui ont permis de générer le processus associé à la ligne de commande ps -eo user,pid,ppid,time,cmd, en partant du processus initial dont le PID est 1.

On remonte les PPID à partir du processus demandé, de PID 5629 : son parent a le PID 5622 ; le parent de 5622 a le PID 5571 ; celui de 5571 a le PID 1611 ; et le parent de 1611 a le PID 1. On remet ensuite cette chaîne dans l'ordre demandé, en partant du processus initial.

RÉPONSE À RETENIR

La succession des PID est **1 → 1611 → 5571 → 5622 → 5629**.

B

Question 6.

Sujet p. 8

> Les trois états principaux d'un processus sont « prêt », « bloqué » et « élu ». Donner la définition de chacun de ces trois états.

Il faut distinguer le fait qu'un processus puisse s'exécuter de son exécution effective : un processus prêt attend le processeur, tandis qu'un processus élu l'utilise. Un processus bloqué attend qu'un événement attendu se produise ou qu'une ressource devienne disponible.

RÉPONSE À RETENIR

- **Prêt :** le processus peut s'exécuter, mais attend que l'ordonnanceur lui attribue le processeur.
- **Bloqué :** le processus ne peut pas s'exécuter tant qu'un événement attendu ne s'est pas produit ou qu'une ressource n'est pas disponible.
- **Élu :** le processus est en cours d'exécution sur le processeur.

Question 7.

Sujet p. 8

> On considère trois processus avec un quantum fixé à 2 unités de temps : P_1 a un temps d'exécution de 5 et un instant d'arrivée de 0 ; P_2 a un temps d'exécution de 3 et un instant d'arrivée de 1 ; P_3 a un temps d'exécution de 4 et un instant d'arrivée de 5. Recopier et compléter le chronogramme d'exécution des processus selon l'ordonnancement par tourniquet.

Le quantum vaut 2 unités de temps. On représente les intervalles sous la forme [début, fin[. P_1 s'exécute d'abord de 0 à 2. P_2, arrivé à 1, est alors le seul autre processus prêt et s'exécute de 2 à 4 ; il lui reste une unité. P_1 s'exécute ensuite de 4 à 6. P_3 arrive à 5 et rejoint la file d'attente. À 6, P_2 est en tête, puis viennent P_3 et P_1. On applique ensuite le même principe jusqu'à la fin des trois processus.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Intervalle de temps	Processus exécuté	Temps restant à la fin
[0, 2[	P_1	3
[2, 4[	P_2	1
[4, 6[	P_1	1
[6, 7[	P_2	0
[7, 9[	P_3	2
[9, 11[	P_1	0
[11, 13[	P_3	0

Chronogramme : **0–2 : P_1 ; 2–4 : P_2 ; 4–6 : P_1 ; 6–7 : P_2 ; 7–9 : P_3 ; 9–11 : P_1 ; 11–13 : P_3.**

C

Question 8.

Sujet p. 9

> Déterminer la table de vérité de l'expression booléenne NON (a ET b).

La valeur de `a ET b` est 1 uniquement lorsque a et b valent tous deux 1. L'opérateur NON inverse cette valeur : `NON (a ET b)` vaut donc 0 uniquement lorsque a = 1 et b = 1.

RÉPONSE À RETENIR

a	b	a ET b	NON (a ET b)
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Question 9.

Sujet p. 9

> À l'aide d'une deuxième table de vérité, justifier que NON (a ET b) = (NON a) OU (NON b).

On calcule les deux membres de l'égalité pour les quatre combinaisons possibles de a et b. Comme les résultats des deux expressions sont égaux à chaque ligne, elles sont équivalentes.

RÉPONSE À RETENIR

a	b	NON a	NON b	(NON a) OU (NON b)	NON (a ET b)
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Les deux dernières colonnes étant identiques, **NON (a ET b) = (NON a) OU (NON b)**.

Exercice 3

4 points

Programmation orientée objet et arbres binaires de recherche

A

Question 1.

Sujet p. 10

> Recopier et compléter les lignes de code numérotées de 2 à 5 de la classe Personnage.

Le constructeur reçoit les caractéristiques du personnage. `self` désigne l'instance en cours de création ; les affectations enregistrent les paramètres dans ses attributs.

RÉPONSE À RETENIR

```
def __init__(self, nom_clan, pts_vie, pts_force):
    self.clan = nom_clan
    self.vie = pts_vie
    self.force = pts_force
```

Question 2.

Sujet p. 10

> Donner les attributs ainsi que les méthodes de cette classe.

Les attributs contiennent les données propres à chaque personnage. Les méthodes sont les fonctions définies dans la classe.

RÉPONSE À RETENIR

Attributs : `clan`, `vie` et `force`.

Méthodes : `__init__` (constructeur), `attaque` et `defense`.

Question 3.

Sujet p. 10

> Écrire l'instruction qui permet d'instancier le personnage luther du clan « Umbrella » avec 100 points de vie et 15 points de force.

On instancie la classe `Personnage` en passant au constructeur le clan, le nombre de points de vie, puis le nombre de points de force.

RÉPONSE À RETENIR

```
luther = Personnage("Umbrella", 100, 15)
```

Question 4.

Sujet p. 10, 11

> Écrire le code Python de la méthode attaque correspondant à la description suivante : elle s'applique sur un personnage (l'attaquant) et prend en paramètre autre_perso (le personnage attaqué) ; elle abaisse le nombre de points de vie du personnage attaqué d'une valeur égale au nombre de points de force du personnage qui attaque ;

FormaV - document d'entraînement non officiel

> elle renvoie 1 s'il reste des points de vie au personnage attaqué, 0 sinon. L'exemple indique qu'une attaque de luther, qui a 15 points de force, contre rayleigh, qui a 120 points de vie, met à jour ses points de vie à 105 et renvoie 1.

Dans la méthode, `self` désigne l'attaquant. On retranche donc sa force aux points de vie de `autre_perso`. Le personnage attaqué reste actif si ses points de vie sont strictement positifs : la méthode renvoie alors 1, et renvoie 0 sinon.

Dans l'exemple, les points de vie de rayleigh passent de 120 à 120 − 15 = 105 ; la méthode renvoie donc 1.

RÉPONSE À RETENIR

```
def attaque(self, autre_perso):
    autre_perso.vie = autre_perso.vie - self.force
    if autre_perso.vie > 0:
        return 1
    else:
        return 0
```

Question 5.

Sujet p. 11

> Le développeur souhaite garder en mémoire toutes les actions réalisées par un personnage afin de pouvoir les annuler les unes après les autres et ainsi revenir à un état antérieur du personnage. Indiquer, sans justifier, la structure de données la plus pertinente à utiliser pour garder en mémoire les différentes actions du personnage.

Comme les actions doivent être annulées dans l'ordre inverse de leur réalisation, la structure adaptée suit le principe « dernier entré, premier sorti ».

RÉPONSE À RETENIR

Une **pile**.

B

Question 6.a)

Sujet p. 11

> Pour l'arbre binaire de recherche représenté, donner sans justification la hauteur et la taille de l'arbre.

La hauteur est comptée en niveaux : l'énoncé précise qu'un arbre réduit à un seul nœud a une hauteur égale à 1. Le plus long chemin de la racine à une feuille comprend ici trois nœuds. La taille correspond au nombre total de nœuds représentés.

RÉPONSE À RETENIR

Hauteur : 3. **Taille :** 6.

Question 6.b)

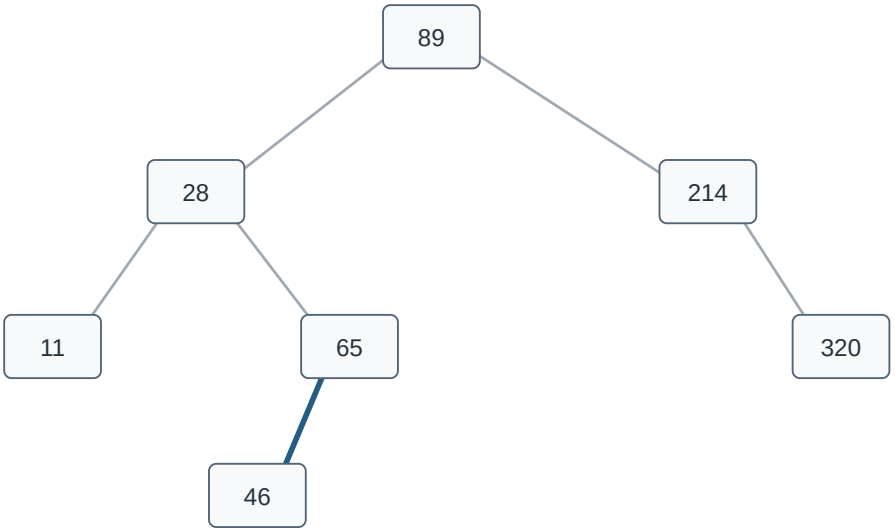
Sujet p. 11, 12

> Insérer dans l'arbre binaire de recherche représenté la clé dont la valeur est 46. Représenter l'arbre obtenu après cette insertion.

Pour insérer 46, on compare sa valeur à celle des nœuds rencontrés : 46 < 89, donc on va à gauche ; 46 > 28, donc on va à droite ; 46 < 65, donc on l'insère comme fils gauche de 65, dans la place vide.

FormaV - document d'entraînement non officiel

Arbre binaire de recherche après l'insertion de 46



Le nœud 46 est le fils gauche de 65.

> **RÉPONSE À RETENIR**
>
> L'arbre obtenu après l'insertion est représenté ci-dessous. L'arête mise en évidence relie 65 au nouveau nœud 46.

Question 7.a)

Sujet p. 12

> Pour la fonction parcours fournie, donner le nom de ce parcours.

La fonction parcourt le sous-arbre gauche, affiche la racine, puis parcourt le sous-arbre droit. C'est la définition du parcours infixe.

> **RÉPONSE À RETENIR**
>
> **Nom du parcours :** parcours infixe (ou parcours en ordre).

Question 7.b)

Sujet p. 11, 12

> En considérant l'arbre donné en exemple, sans l'ajout de la valeur 46, indiquer l'ordre dans lequel les valeurs des nœuds seront affichées par la fonction parcours.

On considère l'arbre initial représenté à la question 6, sans l'ajout de 46. Le parcours infixe affiche d'abord les nœuds du sous-arbre gauche par ordre croissant, puis la racine, puis ceux du sous-arbre droit par ordre croissant.

> **RÉPONSE À RETENIR**
>
> **Ordre d'affichage :** 11, 28, 65, 89, 214, 320.

FormaV - document d'entraînement non officiel

Question 8.

Sujet p. 11, 12, 13

> Pour le jeu vidéo défini dans la partie A, on souhaite collecter l'équipement d'un personnage à l'aide d'un arbre binaire de recherche. Chaque type de matériel est identifié par un entier et un personnage peut en posséder plusieurs exemplaires. Les clés sont des tableaux non dynamiques contenant deux entiers : la référence du matériel en première position et la quantité correspondante en deuxième position ; le placement dans l'arbre se fait selon la référence. Recopier et compléter la fonction recherche_dichotomique(arbre, ref_materiel), qui prend en paramètre un arbre binaire de recherche et un entier (référence du matériel), et renvoie la quantité associée si le matériel est présent dans l'arbre, -1 sinon.

Chaque nœud contient une clé de la forme `[référence, quantité]`. La recherche compare la référence demandée à l'élément d'indice 0 de la clé à la racine. Si elle est plus petite, on poursuit dans le sous-arbre gauche ; si elle est plus grande, dans le sous-arbre droit. Si elle est égale, la quantité recherchée se trouve à l'indice 1. Lorsque l'arbre devient vide, la référence est absente et la fonction renvoie −1.

Trace avec les données de l'arbre d'équipement du sujet : pour la référence 28, on compare d'abord à 89, puis on va à gauche et on trouve 28 : l'élément d'indice 1 vaut 3, donc la fonction renvoie 3. Pour la référence 46, on compare successivement à 89, 28 et 65 ; à chaque étape, on choisit le sous-arbre correspondant à la comparaison. Le sous-arbre gauche de 65 est vide : la fonction renvoie −1.

À chaque appel récursif, la recherche descend dans un sous-arbre ; elle s'arrête donc lorsqu'elle trouve la référence ou atteint un arbre vide. Elle examine au plus un nœud par niveau : son coût est proportionnel à la hauteur de l'arbre, au plus à sa taille.

RÉPONSE À RETENIR

```
def recherche_dichotomique(arbre, ref_materiel):
    if est_vide(arbre):
        return -1
    elif ref_materiel < valeur_racine(arbre)[0]:
        return recherche_dichotomique(gauche(arbre), ref_materiel)
    elif ref_materiel > valeur_racine(arbre)[0]:
        return recherche_dichotomique(droit(arbre), ref_materiel)
    else:
        return valeur_racine(arbre)[1]
```

Sur l'arbre d'équipement du sujet, la recherche de 28 renvoie **3** et celle de 46 renvoie **−1**.

Sujet : https://www.sujetdebac.fr/annales-pdf/2023/spe-numerique-informatique-2023-amerique-nord-1-sujet-officiel.pdf

© FormaV EI • Proposition de correction, distincte d'un corrigé officiel.

FormaV - document d'entraînement non officiel

Copyright © 2027 FormaV®. Tous droits réservés.

FormaV® est une marque déposée auprès de l'INPI sous le numéro 5227897.

Ce document a été élaboré par FormaV® avec le plus grand soin afin d'accompagner chaque apprenant vers la réussite de ses examens. Son contenu (textes, graphiques, méthodologies, tableaux, exercices, concepts, mises en forme) constitue une œuvre protégée par le droit d'auteur.

Toute copie, partage, reproduction, diffusion ou mise à disposition, même partielle, gratuite ou payante, est strictement interdite sans accord préalable et écrit de FormaV®, conformément aux articles L.111-1 et suivants du Code de la propriété intellectuelle. Dans une logique anti-plagiat, FormaV® se réserve le droit de vérifier toute utilisation illicite, y compris sur les plateformes en ligne ou sites tiers.

En utilisant ce document, vous vous engagez à respecter ces règles et à préserver l'intégrité du travail fourni. La consultation de ce document est strictement personnelle.

Merci de respecter le travail accompli afin de permettre la création continue de ressources pédagogiques fiables et accessibles.