

Institut Supérieur du Numérique

Examen de Structures de données et complexité Algorithmiques L2 S3 2025-2026

Durée : 1h30

Aucun document autorisé

Le sujet est constitué de 2 pages

Exercice 1 : (5 points)

1. Donner une implémentation (structure + algorithmes) d'une **file (queue)** en utilisant **deux piles**. **Indication** : on peut utiliser une pile pour enfiler dans la file (enqueue) et une deuxième pour défiler (dequeue)
2. Donner la complexité de l'opération défiler pour une file de n élément ?

Exercice 2 (5 points)

On insère dans un arbre binaire de recherche vide les valeurs suivantes dans cet ordre :
40, 20, 10, 30, 60, 50, 70, 65

1. Dessiner l'arbre obtenu.
2. Donner le parcours **largeur (BFS)** de l'arbre.
3. Donner le parcours en profondeur **préfixe** de l'arbre.
4. Écrire une fonction qui calcule la **hauteur** d'un arbre binaire.
5. Écrire une fonction qui trouve l'élément le plus petit dans un arbre binaire de recherche.
Donnez sa complexité en $O(n)$.

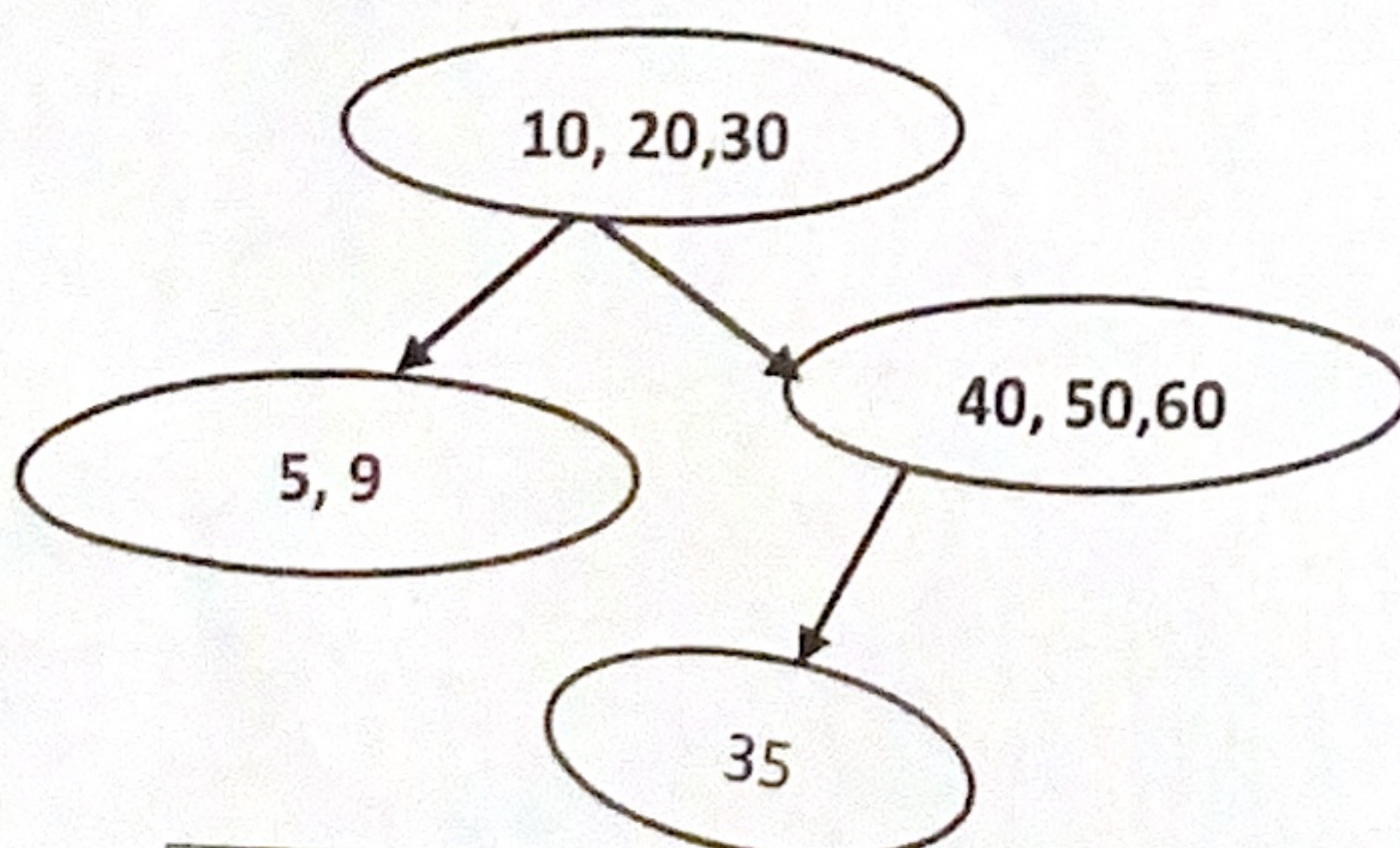
Exercice 3 (10 points)

On souhaite implémenter un arbre binaire de recherche avec les propriétés suivantes :

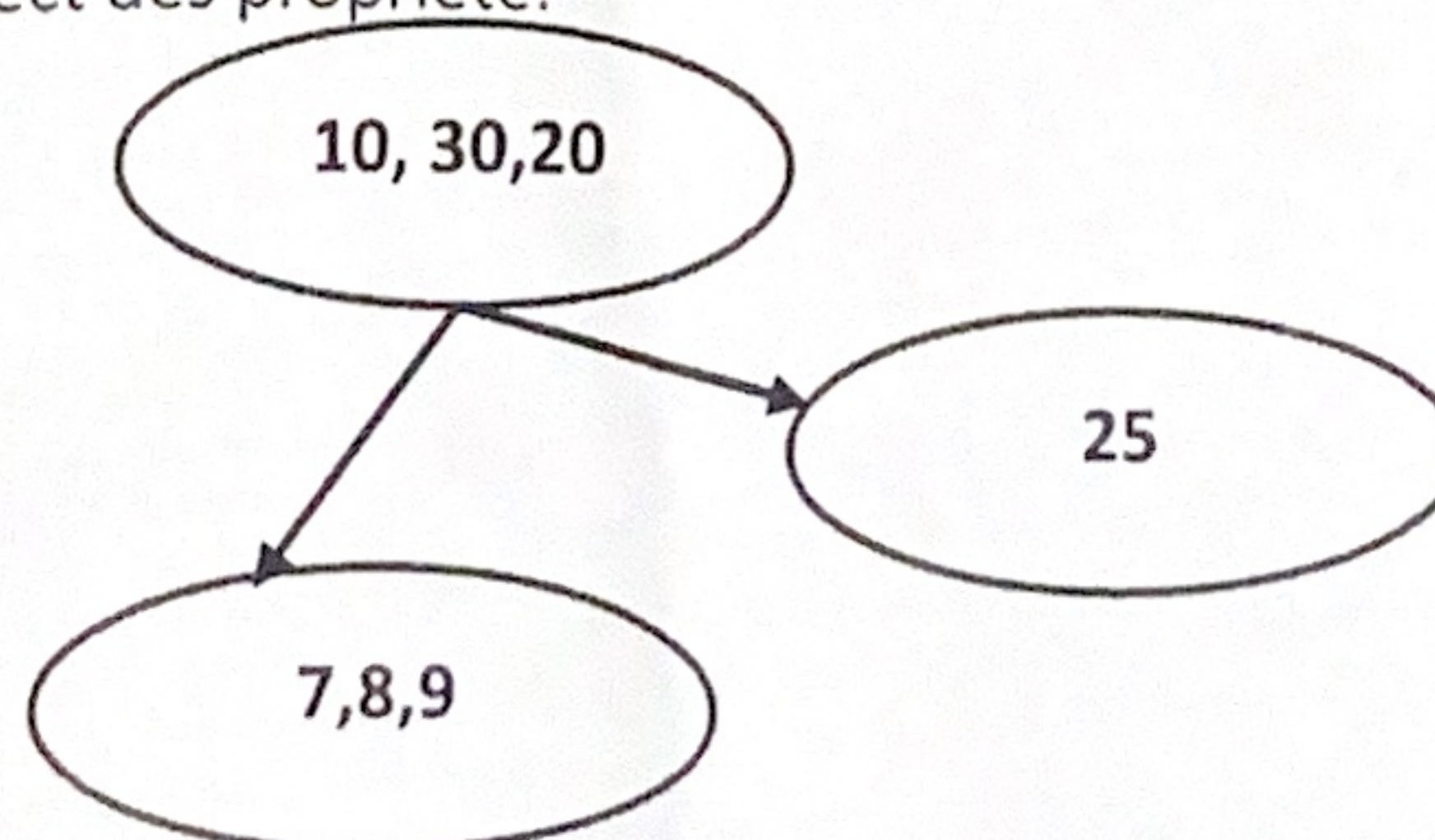
- 1- Chaque nœud contient au maximum trois valeur dans une liste chaînées
- 2- Les valeurs dans chaque nœud sont dans l'ordre croissant
- 3- La première valeur de chaque nœud est plus grande des values dans le sous arbre gauche ; La dernière valeur est inférieure des values dans le sous arbre droit

L'objectif est d'implémenter cette structure en répondant aux questions suivantes.

1. Lequel des arbres suivants respectent les propriétés citées ci-dessous ? Propose une correction en cas de non-respect des propriété.



a)



b)

2. Écrire la structure correspondantes (on suppose qu'il existe une structure liste avec les Operations nécessaires) ?
3. Écrire une fonction qui prend comme paramètres un nœud et vérifie si les valeurs qu'elles contiennent (s'il y en a) sont tirés selon l'ordre croissant ?
4. Écrire une fonction qui vérifie si les valeurs directes à gauche et celles immédiatement à droites respectent les propriétés précédentes
5. Écrire une fonction qui vérifie si un arbre donné en paramètre vérifie les propriétés précédentes (Indication : une fonction récursive qui peut utiliser les 2 fonctions en 2 et 3)
6. Écrire une fonction qui recherche si un élément e, donnée est dans un arbre B, donné également en paramètre.
7. Écrire une fonction qui ajoute un élément dans un arbre binaire. **Indication** : Si la racine n'est pas encore remplie, on lui ajoute l'élément dans le bon ordre, sinon on l'ajoute dans son sous arbre gauche ou droit selon les propriétés citées précédemment.