TD5 - Structure de données

Code	LU2IN006
Type de cours	Travaux dirigés
Complété ?	☐
Jour du cours	@20/02/2023

`PINHO FERNANDES Enzo - L2 Mono-info - GR6`

TD5 :

▼ Exercice 1 : Rappels sur les tas

📖 **Enoncé :**

Exercice 1 – Rappels sur les tas

Q 1.1 Donner la définition de la structure de données abstraite appelée “tas”.

Q 1.2 En utilisant uniquement l’opération `swap(pere, fils)`, construire le tas obtenu par insertion successives des entiers $[10, 2, 5, 4, 7, 15, 1, 3]$, la valeur de ces entiers servant directement de clé.

Q 1.3 De la même manière, quel arbre obtient-on ensuite en supprimant l’élément de plus petite clé ? Détailler les étapes.

Q 1.4 Combien d’éléments peut-on avoir dans un tas de hauteur h (en nombre d’arcs) ?

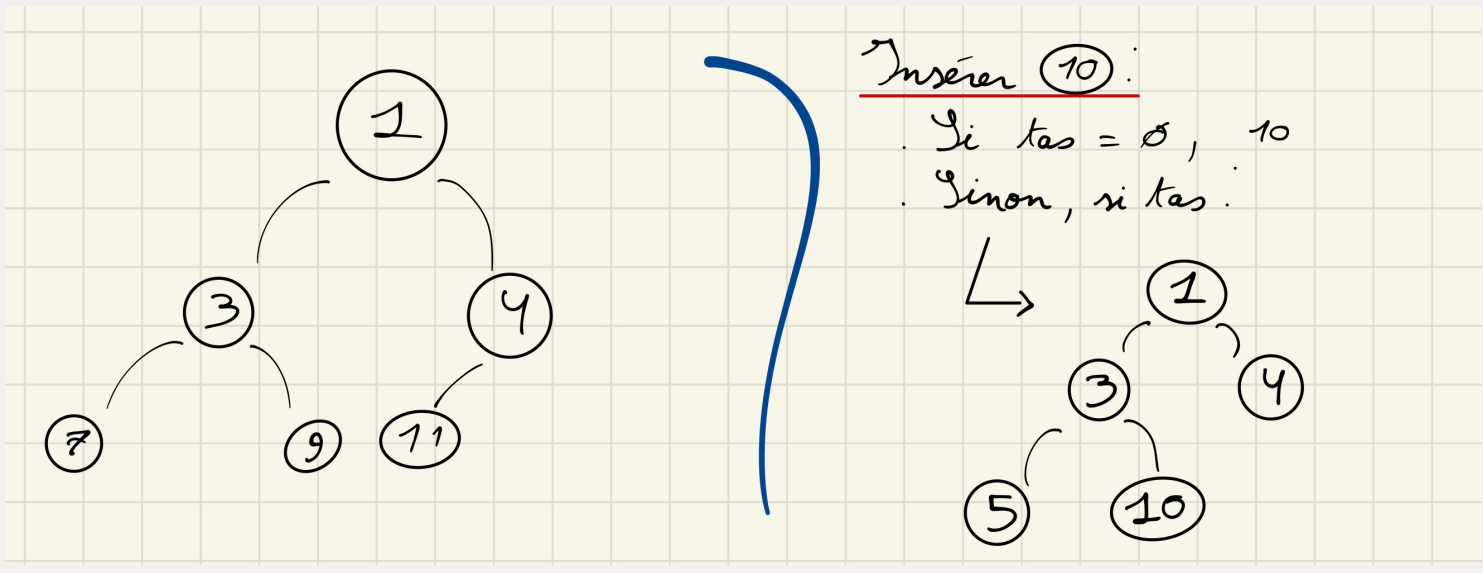
Q 1.5 Étant donné n éléments à stocker dans un tas, écrire h, la hauteur du tas, en fonction de n.

Q 1.6 Quelle est la complexité-temps pire des cas d’insertion d’un élément dans un tas contenant n éléments ? Même question pour la suppression de l’élément de plus petite clé ?

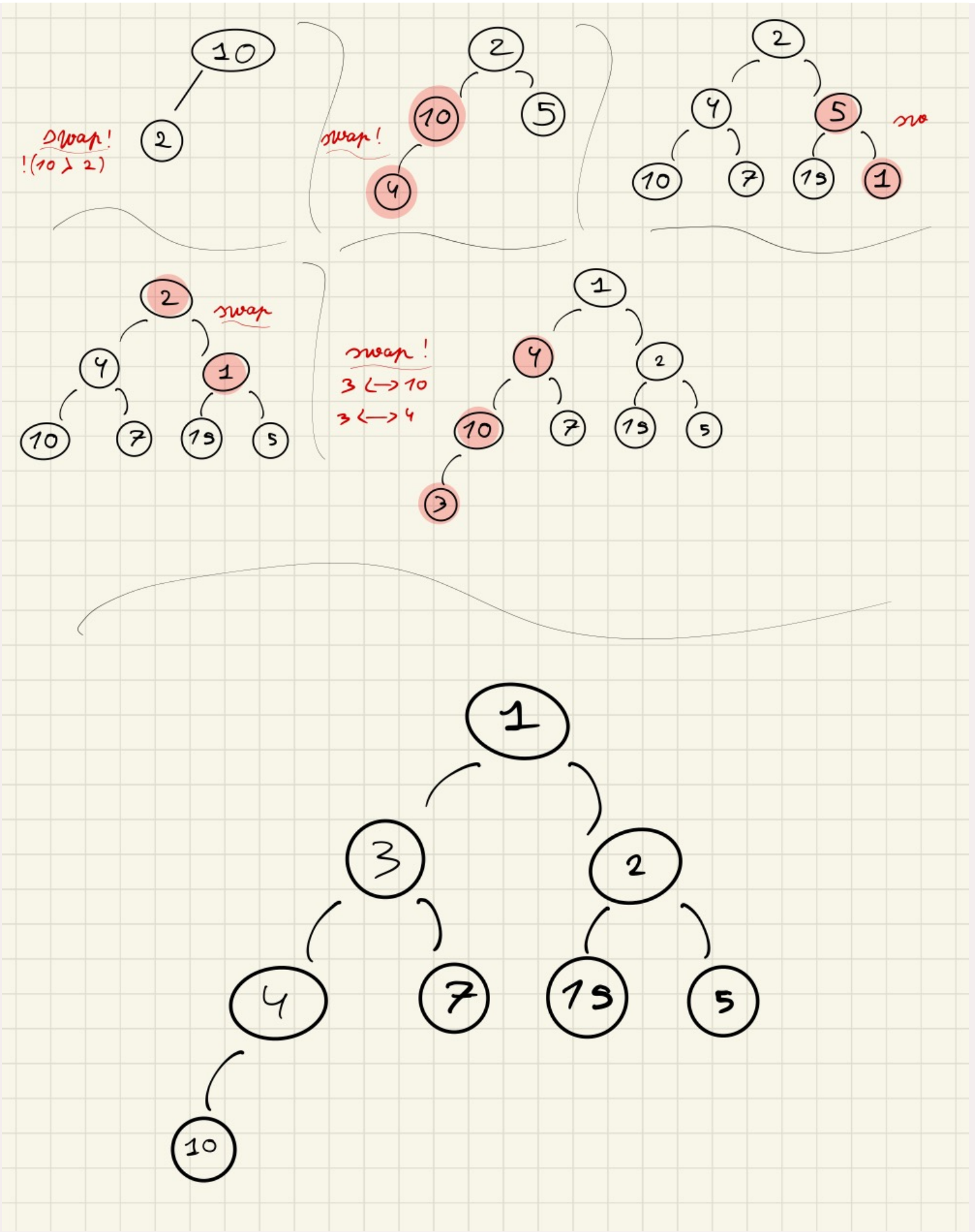
Réponses :

▼ Question 1 :

- Un tas est un arbre binaire, (presque) complet et tassé à gauche.
- `∀ noeud, ∀ enfants de noeud, valeur(enfant) > valeur(noeud)`

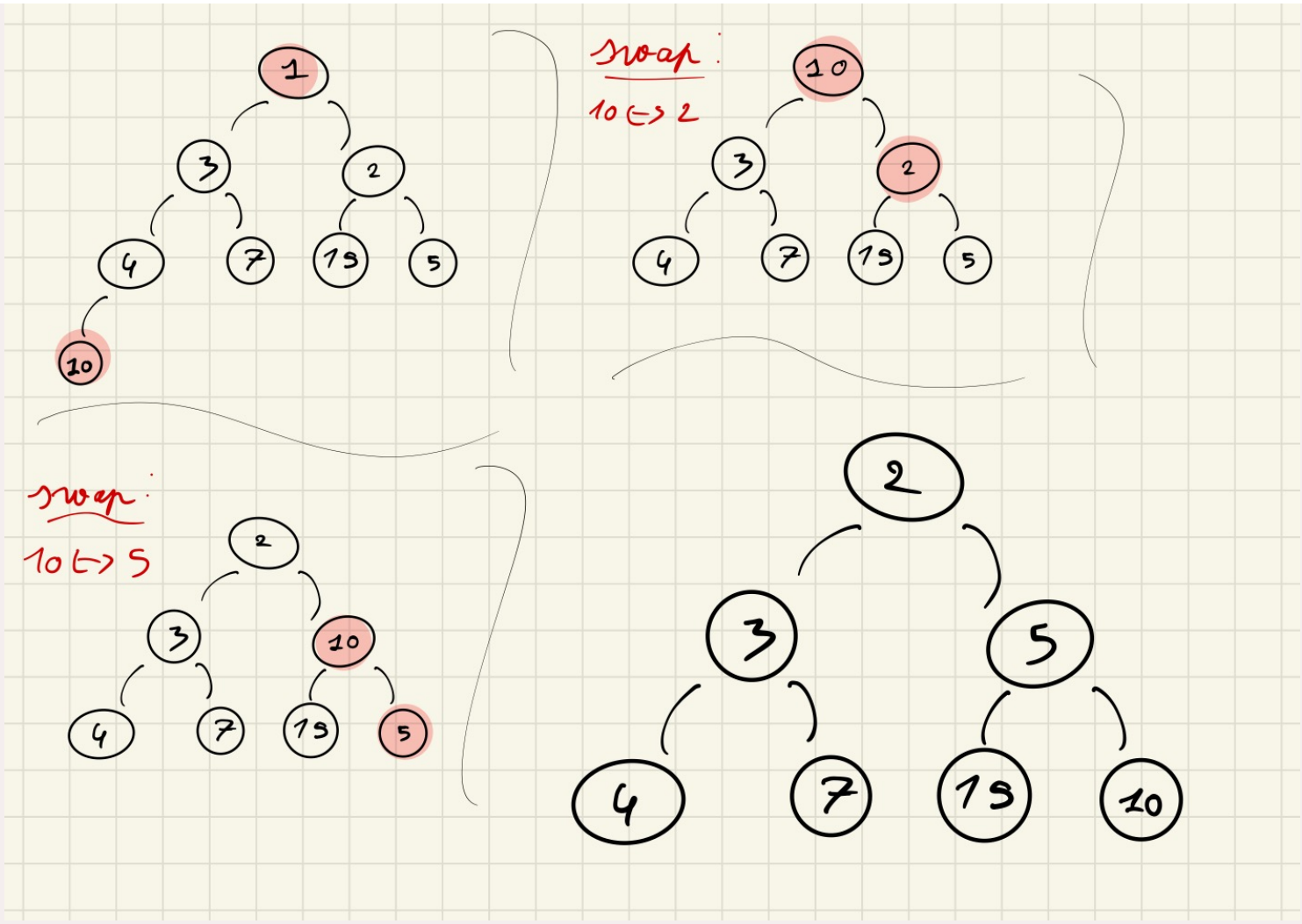


▼ Question 2 :



▼ **Question 3 :**

- ***Supprimer la racine :***
  - On la remplace par le « dernier élément »
  - On « swap » jusqu’à l’équilibre



▼ **Question 4 :**

- $h = 0 \qquad 0 \leq n < 2$
- $h = 1 \qquad 2 \geq n > 4$
- $h = 2 \qquad 4 \geq n > 8$
- h **quelconque :** $2^h \geq n > 2^{h+1}$

▼ **Question 5 :**

- $h = \lfloor log_2(n) \rfloor$
- **Preuve :** $2^h \geq n > 2^{h+1}$
  - Alors $h \geq log_2(n) > h + 1$
  - Alors $h = \lfloor log_2(n) \rfloor$

▼ **Question 6 :**

- **Le pire cas d'insertion :** on « swap » jusqu'à la racine.
  - $h - 1$ swaps $= log_2(n) - 1$ swaps = $O(log_2(n))$
- **Formule en vrac :**
  - $log_2(n) = \frac{ln(n)}{ln(2)}$
  - $log_{10}(n) = \frac{ln(n)}{ln(10)}$
  - $log_{10}(n) = log_2(n) * \frac{ln(2)}{ln(10)} = \frac{ln(n)}{ln(2)} * \frac{ln(2)}{ln(10)} = \frac{ln(n)}{ln(10)}$
- **Le pire cas de suppression de racine :** Même raisonnement, $O(log_2(n))$

▼ Exercice 2 : Tas ternaire

📖 **Enoncé :**

Exercice 2 – Tas ternaire

Afin de réduire la hauteur de l'arbre sous-jacent, on propose d'utiliser un tas ternaire à la place d'un tas binaire. Dans ce type de tas, chaque noeud possède au plus trois fils. La valeur de chaque noeud est inférieure à celle de ses fils.

Q 2.1 Représentez le tas ternaire construit à partir des éléments $[3, 6, 1, 13, 17, 18, 2]$.

Q 2.2 Étant donné n éléments à stocker dans un tas ternaire, écrire la hauteur h du tas en fonction de n. Est-ce que l'utilisation d'un tas ternaire améliore le coût d'insertion dans le pire des cas par rapport à un tas binaire ?

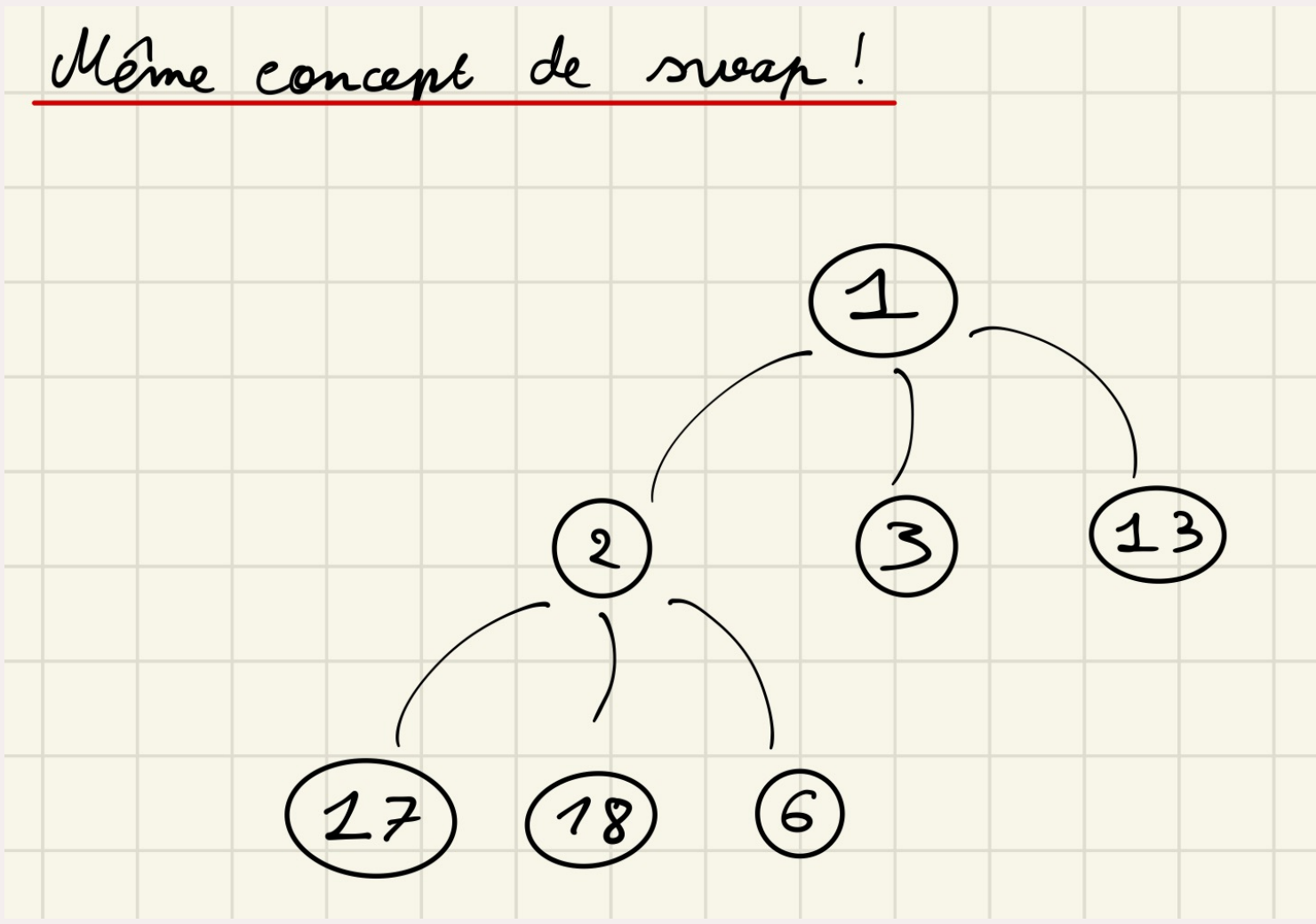
Q 2.3 En numérotant à partir de 1 les nœuds de l'arbre du haut vers le bas, niveau après niveau, et de la gauche vers la droite, quelles relations peut-on établir entre le numéro d'un nœud et ceux de ses trois fils ? Sachant que l'arbre est complet, décrire comment stocker un tas ternaire dans un tableau et écrire les fonctions nécessaires à sa gestion.

Q 2.4 En utilisant les relations définies précédemment, écrire une fonction qui supprime et retourne le plus petit élément d'un tas ternaire contenant des entiers.
Note Afin de maintenir une structure d'arbre ternaire tassé à gauche, il faut remplacer la racine par la feuille qui se trouve à l'extrémité droite.

Q 2.5 Quelle est la complexité-temps pire cas de la suppression d'un élément dans un tas ternaire contenant n éléments.

Réponses :

▼ **Question 1 :**



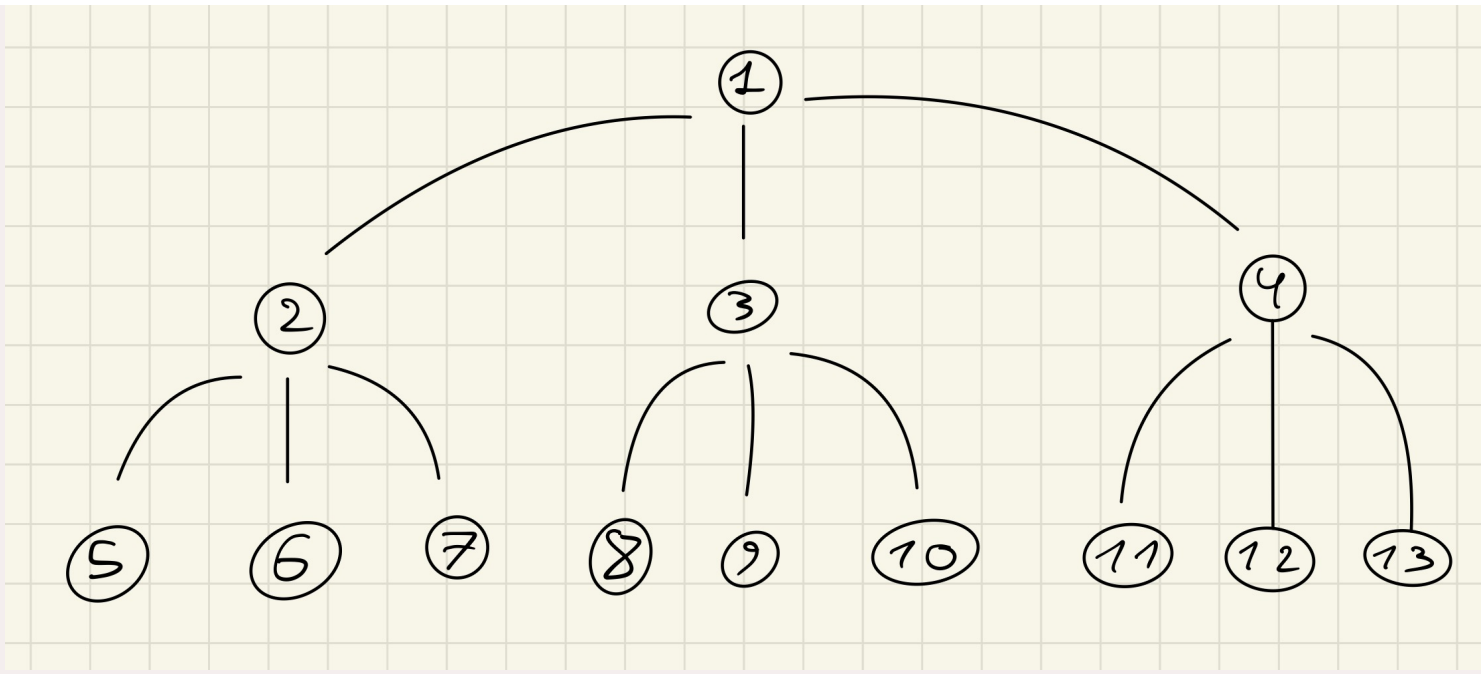
▼ **Question 2 :**

- $0 \leq n \leq 1 : h = 0$
- $2 \leq n \leq 4 : h = 1$
- $5 \leq n \leq 13 : h = 2$
- h **quelconque :** $\Sigma_{k=0}^{h-1} 3^h + 1 \leq n \leq \Sigma_{k=0}^{h} 3^h$
  - $\frac{3^{h+1}-1}{2}$ = la somme à droite de la formule
  - Alors $\frac{3^h-1}{2} + 1 \leq n \leq \frac{3^{h+1}-1}{2}$
    - Alors $3^h \leq 2n + 1 \leq 3^{h+1}$
    - Alors $h < log_3(2n+1) \leq h+1$
    - Alors $h \leq log_3(3n+1) - 1 < h+1$
- $h = \lfloor log_3(2n+1) - 1 \rfloor$

- **Complexité pire-cas insertion :**
  - On aura $h - 1$ opérations, donc $\lfloor log_3(2n+1) - 1 \rfloor - 1$ opérations.
  - $O(log_3(n))$
    - Vu que $O(log_3(n)) = O(log_2(n))$, la complexité est la même.

▼ **Question 3 :**

- Le noeud n a pour enfants : $(3n-1, 3n, 3n+1)$




```
typedef struct tas{
  int size_max;
  int size;
  int *t;
} Tas;

int enfant(Tas *tas, int pos, int num){
  // Tester que la position des enfants n'est pas < tas.size

  return 3*pos-1+num;
}
```

▼ Question 4 :

```
void tasser(Tas *tas, int pos){
  int enfant0_idx = enfant(tas, pos, 0);
  int enfant1_idx = enfant(tas, pos, 1);
  int enfant2_idx = enfant(tas, pos, 2);

  //Ici, on suppose que les enfants existent.
  if(tas->t[enfant0_idx] < tas->t[pos]){
    swap(tas->t, pos, enfant0_idx);
    return tasser(tas, enfant0_idx);
  }

  if(tas->t[enfant1_idx] < tas->t[pos]){
    swap(tas->t, pos, enfant1_idx);
    return tasser(tas, enfant1_idx);
  }

  if(tas->t[enfant2_idx] < tas->t[pos]){
    swap(tas->t, pos, enfant2_idx);
    return tasser(tas, enfant2_idx);
  }
}

// On échange la racine et le dernier élément, on supprime la racine et on applique cette fonction.
```

▼ Question 5 :

- Même chose que la question 2 de cette exercice.