Sujet Zéro 1

Exercice 3

Question 1 La *taille* de l'arbre est égale au nombre de nœuds et vaut 9. La *hauteur* correspond au nombre de nœuds traversés entre la racine (incluse) et la plus éloignée des feuilles (incluse). Elle vaut 4.

Question 2.1 Avant de déterminer le code de G, il faut déterminer le code de son parent D.

- D est enfant droit de $D : 10$ donc son code se termine par 1. Donc $G : 101$.
- G est enfant gauche de $D : 101$ donc son code se termine par 0. Donc $D : 1010$.

Question 2.2 Commençons par convertir 13 en binaire.

1ère méthode : les divisions euclidiennes

- 13 ÷ 2 = 6 et reste 1
- 6 ÷ 2 = 3 et reste 0
- 3 ÷ 2 = 1 et reste 1
- 1 ÷ 2 = 0 et reste 1

le codage binaire se retrouve en relevant les restes en commençant par la fin :

$$13_{10} = 1101_2$$

2ème méthode : la décomposition en puissance de 2

$$13 = 8 + 4 + 1 = \mathbf{1}\times 2^3 + \mathbf{1}\times 2^2 + \mathbf{0}\times 2^1 + \mathbf{1}\times 2^0$$

ce qui s'écrit aussi :

bit	3	2	1	0
puissance	2^3	2^2	2^1	2^1
puissance	8	4	2	1
13	1	1	0	1

Le nœud cherché a pour code 1101. C'est donc l'enfant droit de $F : 110$. C'est donc le nœud I.

Question 2.3 On remarque que :

- le nœud de hauteur 1 se code sur 1 bit
- le nœud de hauteur 2 se code sur 2 bit
- le nœud de hauteur 3 se code sur 3 bit
- le nœud de hauteur 4 se code sur 4 bit
- …
- le nœud de hauteur h se code sur h bit

Ce résultat se montre par récurrence :

1. **Hypothèse** : *les nœuds de hauteurs h sont numérotés sur h bits.*
2. **Cas de base** : soit $h = 1$. L'hypothèse est **vraie** puisque le nœud de hauteur 1 se code sur 1 bit.
3. **Cas récursif** :
   - Supposons que l'*hypothèse soit vraie pour les nœuds de hauteur h* et montrons qu'elle est vraie pour les nœuds de hauteurs $h + 1$.
   - Soit N un nœud de hauteur $h + 1$. Que N soit un enfant gauche ou un enfant droit, il s'écrit avec 1 bit de plus que son parent.
   - Or le parent est un nœud de hauteur h, donc *Par hypothèse* : le parent s'écrit avec h bits.
   - Donc N s'écrit sur $h + 1$ bits.
   - Donc l'*hypothèse soit vraie pour les nœuds de hauteur $h + 1$.*

Question 2.4 Soit un arbre de hauteur *h* et de taille *n*.

1. Puisque la hauteur est le nombre de nœud du plus long chemin entre la racine et une feuille, l'arbre possède plus de nœuds que la hauteur. Donc

$$h \leq n.$$

2. Montrons la deuxième inégalité en utilisant la question précédente.
   - Le nombre de nœuds n est inférieure ou égal à la numérotation de sa feuille la plus à droite.
   - Or cette numérotation est codée sur h bits puisque la feuille est de hauteur h.
   - Donc n est inférieure ou égal au plus nombre codable sur h bits : $n \leq \overbrace{11\ldots111}^{\text{h bits}}$
   - Ce qui donne :

$$\begin{aligned}
n &\leq \overbrace{11\ldots111}^{\text{h bits}}{}_2 \\
n &\leq \overbrace{11\ldots111}^{\text{h bits}}{}_2 + 1 - 1 \\
n &\leq 1\overbrace{00\ldots000}^{\text{h bits}}{}_2 - 1 \\
n &\leq 2^h - 1
\end{aligned}$$

On a bien montré l'inégalité de l'énoncé.

Question 3.1 On obtient le tableau suivant :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Question 3.2 Le parent du nœud i a pour indice le quotient de la division euclidienne de i par 2 : `i // 2`.

Question 4

Voici une proposition avec une boucle `while`.

[13]:
```
def recherche(arbre: list, element):
    n = arbre[0]

    # BOUCLE
    # invariant
    #  -> element n'appartient pas à arbre[1 .. i-1]
    #
    # initialisation
    #  -> i = 1
    i = 1

    # condition d'arrêt
    #  -> i déborde du tableau: i > n
    while not (i > n):
        # si on trouve element, on arrête la recherche
        if arbre[i] == element:
            return True
        i = i + 1

    # si on arrive là, l'invariant est vérifié
    # et tout le tableau a été exploré :
    #  => element n'y est pas
    return False


arbre = [10, 42, 43, 44, None, 46, 47, 48, None, None, 51]
recherche(arbre, 52)
```

[13]: False