

Asie – 2025 – sujet1 - Correction

Exercice 2 (6 points)

Partie A – Coder du texte

1. Taille du texte `txt` = "SIX ANANAS"

👉 *Commentaire :*

En ASCII Latin-1, *chaque caractère est codé sur 1 octet (8 bits)*.

La chaîne "SIX ANANAS" contient :

- S I X → 3 caractères
- ESPACE → 1 caractère
- A N A N A S → 6 caractères
- Total : **10 caractères**

Réponse :

- Taille en octets : **10 octets**
- Taille en bits : **$10 \times 8 = 80$ bits**

2. Codage hexadécimal de la chaîne `txt`

On utilise la table ISO/CEI 8859-1.

Caractère	S	I	X	SP (espace)	A	N
Code hexadécimal	53	49	58	20	41	4E

Réponse :

53 49 58 20 41 4E 41 4E 41 53

👉 *Commentaire :* Chaque caractère est codé indépendamment sur **8 bits**, ce qui explique l'absence de compression à ce stade.

Partie B – Compression de Huffman

3. Tableau d'occurrences de txt

Texte :

S I X A N A N A S

Symbole	S	I	X	SP	A	N
Nombre d'occurrences	2	1	1	1	3	2

👉 *Commentaire : On compte **chaque apparition** d'un symbole, y compris l'espace.*

4. Signification de la somme des occurrences

Somme :

$$2 + 1 + 1 + 1 + 3 + 2 = 10$$

La somme des occurrences correspond **au nombre total de caractères du texte**.

👉 *Commentaire : C'est une vérification simple permettant de contrôler que le tableau d'occurrences est correct.*

5. Fonction Python occurrence

```
def occurrence(texte):  
    dico = {}  
    for lettre in texte:  
        if lettre in dico:  
            dico[lettre] = dico[lettre] + 1  
        else:  
            dico[lettre] = 1  
    return dico
```

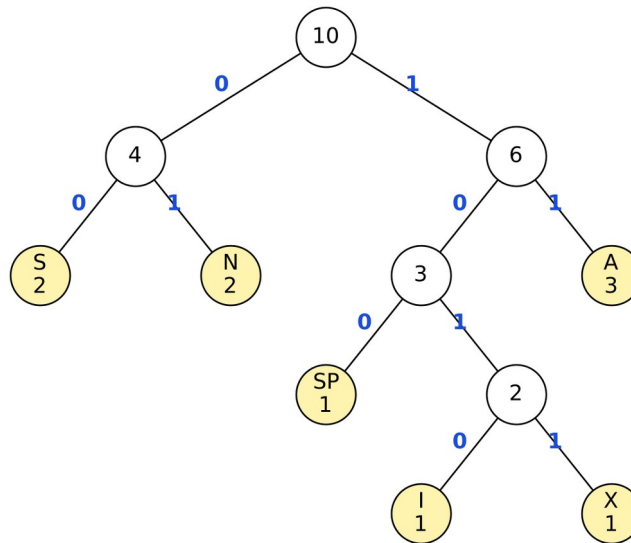
👉 *Commentaire :*

- On parcourt le texte caractère par caractère
 - On incrémente si la clé existe
 - Sinon, on initialise à 1
-

6. Construction de l'arbre de Huffman

On range les arbres par poids croissant (à poids égal, dans l'ordre d'apparition dans le texte) : I(1), X(1), SP(1), S(2), N(2), A(3). À chaque étape, on extrait les deux premiers arbres (le premier extrait à gauche), on les greffe, et le nouvel arbre est inséré **avant** les arbres de poids supérieur ou égal :

- $I(1) + X(1) \rightarrow 2$; file : SP(1), 2, S(2), N(2), A(3)
- $SP(1) + 2 \rightarrow 3$; file : S(2), N(2), 3, A(3)
- $S(2) + N(2) \rightarrow 4$; file : 3, A(3), 4
- $3 + A(3) \rightarrow 6$; file : 4, 6
- $4 + 6 \rightarrow 10$: il ne reste qu'un arbre, c'est l'arbre de Huffman.



👉 *Commentaire : à poids égal, un autre ordre de départ (par exemple SP, I, X) donne un arbre différent mais tout aussi valable : les longueurs des codes, donc la taille du texte codé, restent les mêmes.*

7. Poids de la racine

Poids de la racine : **10**. C'est la somme des nombres d'occurrences, c'est-à-dire le **nombre total de caractères** du texte txt.

8. Type de parcours pour créer la table de codage

Un **parcours en profondeur** (par exemple préfixe) : on descend de la racine vers chaque feuille en ajoutant 0 à chaque passage à gauche et 1 à chaque passage à droite.

👉 *Commentaire : à l'arrivée sur une feuille, le chemin parcouru est le code du symbole.*

9. Table de codage de Huffman

On lit les bits sur le chemin de la racine à chaque feuille de l'arbre de la question 6 :

Symbole	S	N	SP	I	X	A
Code	00	01	100	1010	1011	11

⚠ Remarque : selon l'ordre choisi à poids égal, d'autres tables sont possibles, toutes valables tant que :

- aucun code n'est le début d'un autre (code préfixe), ce qui permet de décoder sans séparateur ;
- les symboles fréquents ont des codes plus courts (ou égaux).

10. Pourquoi le code de Huffman est un code de longueur variable

Les codes n'ont pas tous la même longueur : ici de **2 bits** (S, N, A) à **4 bits** (I, X). Les symboles fréquents sont codés avec peu de bits, les symboles rares avec davantage de bits.

Cela permet de réduire la longueur moyenne du message codé.

11. Codage de txt avec Huffman

Texte :

S I X ESPACE A N A N A S

Codage : 00 1010 1011 100 11 01 11 01 11 00

Message binaire : **0010101011100110111011100**

Longueur : **25 bits** (3 × 2 bits pour A, 2 × 2 pour S, 2 × 2 pour N, 3 bits pour SP, 4 bits pour I et pour X).

12. Taux de compression

Taille initiale (Partie A)

- 80 bits

Taille compressée

- 25 bits

Calcul du taux de compression

$$\text{Taux} = (80 - 25) / 80 = 55 / 80 = 0,6875$$

Taux de compression : 68,75 % (environ 69 %)

👉 *Commentaire : 68,75 % est bien compris entre 20 % et 90 %, ce qui confirme l'affirmation de l'énoncé « On observe ainsi des réductions de taille de l'ordre de 20 % à 90 % ».*