# EXERCICE 2 (6 points)

*Cet exercice porte sur les arbres et la compression d'un fichier texte.*

Quand il s'agit de transmettre de l'information sur un canal non bruité, l'objectif prioritaire est de minimiser la taille de la représentation de l'information : c'est le problème de la *compression de données*. Le code de Huffman (1952) est un code de longueur variable optimal, c'est-à-dire tel que la longueur moyenne d'un texte codé est minimale. On observe ainsi des réductions de taille de l'ordre de 20 % à 90 %. Ce code est largement utilisé, souvent combiné avec d'autres méthodes de compression.

## Partie A : Coder du texte

On donne, en Figure 1 ci-dessous, la table d'encodage hexadécimal des caractères ISO/CEI 8859-1, dite ASCII Latin 1.

| | x0 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | xA | xB | xC | xD | xE | xF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x | *positions inutilisées* | | | | | | | | | | | | | | | |
| 1x | | | | | | | | | | | | | | | | |
| 2x | SP | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4x | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5x | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6x | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7x | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | |
| 8x | *positions inutilisées* | | | | | | | | | | | | | | | |
| 9x | | | | | | | | | | | | | | | | |
| Ax | NBSP | ¡ | ¢ | £ | ¤ | ¥ | ¦ | § | ¨ | © | ª | « | ¬ |  | ® | ¯ |
| Bx | ° | ± | ² | ³ | ´ | µ | ¶ | · | ¸ | ¹ | º | » | ¼ | ½ | ¾ | ¿ |
| Cx | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| Dx | Ð | Ñ | Ò | Ó | Ô | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |
| Ex | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| Fx | ð | ñ | ò | ó | ô | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ |

Figure 1. Table ISO/CEI 88-59-1

Chaque caractère est codé sur 8 bits, soit deux chiffres hexadécimaux, correspondant respectivement à la ligne et à la colonne à l'intersection desquelles il figure.

Par exemple, pour la lettre 'H' figurant à l'intersection de la ligne '4x' et de la colonne 'x8', le code hexadécimal est '48'.

La chaîne de caractère 'Hello_World_!' est codé par :

'48 65 6C 6C 6F 5F 57 6F 72 6C 64 5F 21'

Dans cette table, le caractère ESPACE est symbolisé par SP.
Soit la chaîne de caractères `txt = "SIX ANANAS"`.

1. Calculer la taille en octets du texte contenu dans la variable `txt`. En déduire la taille en bits nécessaire pour le stocker.

2. Donner le codage de la chaîne de caractères `txt`.

## Partie B : Compression de Huffman

### Nombre d'occurrences

On appelle nombre d'occurrences d'un symbole le nombre de répétitions de ce symbole dans le texte étudié. Ainsi, dans la phrase "DEECDDEBFACCECCEDBAEE" on peut associer le tableau d'occurrences ci-dessous :

| Symbole | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| Nombre d'occurrences | 2 | 2 | 5 | 4 | 7 | 1 |

3. Écrire le tableau d'occurrences associé à la chaîne de caractères `txt`.

4. Préciser à quoi correspond la somme des nombres d'occurrences.

Ce tableau d'occurrences peut être stocké dans un dictionnaire Python où les clés sont les symboles rencontrés dans le texte et les valeurs les nombres d'occurrences de chaque symbole. Ainsi, pour l'exemple ci-dessus, le dictionnaire serait `{'D' : 4, 'E' : 7, 'C' : 5, 'B' : 2, 'F' : 1, 'A' : 2}`.

5. Recopier et compléter le code de la fonction `occurrence` ci-dessous qui, pour un texte passé en paramètre, renvoie le dictionnaire d'occurrences associé.

```
1 def occurrence(texte):
2     dico = {}
3     for lettre in ... :
4         if lettre in ... :
5             dico[lettre] = dico[lettre]+1
6         else:
7             ...
8     return ...
```

### Arbre de Huffman

L'algorithme de Huffman met en œuvre plusieurs structures de données. Il opère sur un ensemble dynamique d'arbres binaires pondérés (une *forêt*), structuré en file à priorité.

Initialement, la *forêt* est constituée d'arbres binaires,

- tous restreints à leur seule racine, dont l'étiquette est un symbole du texte ;
- et respectivement dotés d'un poids correspondant à l'effectif de ce symbole.

Une opération de greffe de deux arbres pondérés est possible : l'arbre résultant est un arbre binaire dont :

- la racine est un nœud sans étiquette ;
- les sous-arbres gauches et droits sont les deux arbres greffés ;
- le poids est la somme des poids de ces deux sous-arbres.

La file à priorité, qui contient tous les arbres considérés, est une structure permettant

- l'extraction : le premier arbre disponible est un arbre de priorité maximale parmi tous les arbres ;
- l'insertion : tout nouvel arbre pondéré est inséré
  - après tous ceux qui ont une priorité strictement plus grande que la sienne ;
  - avant tous ceux qui ont une priorité inférieure ou égale à la sienne.

Pour construire l'arbre de Huffman, tant que la *forêt* compte au moins deux arbres,

- les deux arbres prioritaires sont extraits de la file ;
- ils sont greffés en un nouvel arbre pondéré ;
- et celui-ci est inséré dans la file à priorité.

Une fois l'arbre construit, on pondère les arêtes en partant de la racine : 0 pour les arêtes menant aux enfants gauches, 1 pour les arêtes menant aux enfants droits.

Le schéma de la figure ci-dessous indique comment on construit un arbre de Huffman en fonction du tableau d'occurrences.

Pour plus de clarté, les étiquettes de tous les nœuds ont été remplacées par le poids de l'arbre dont ils sont la racine.

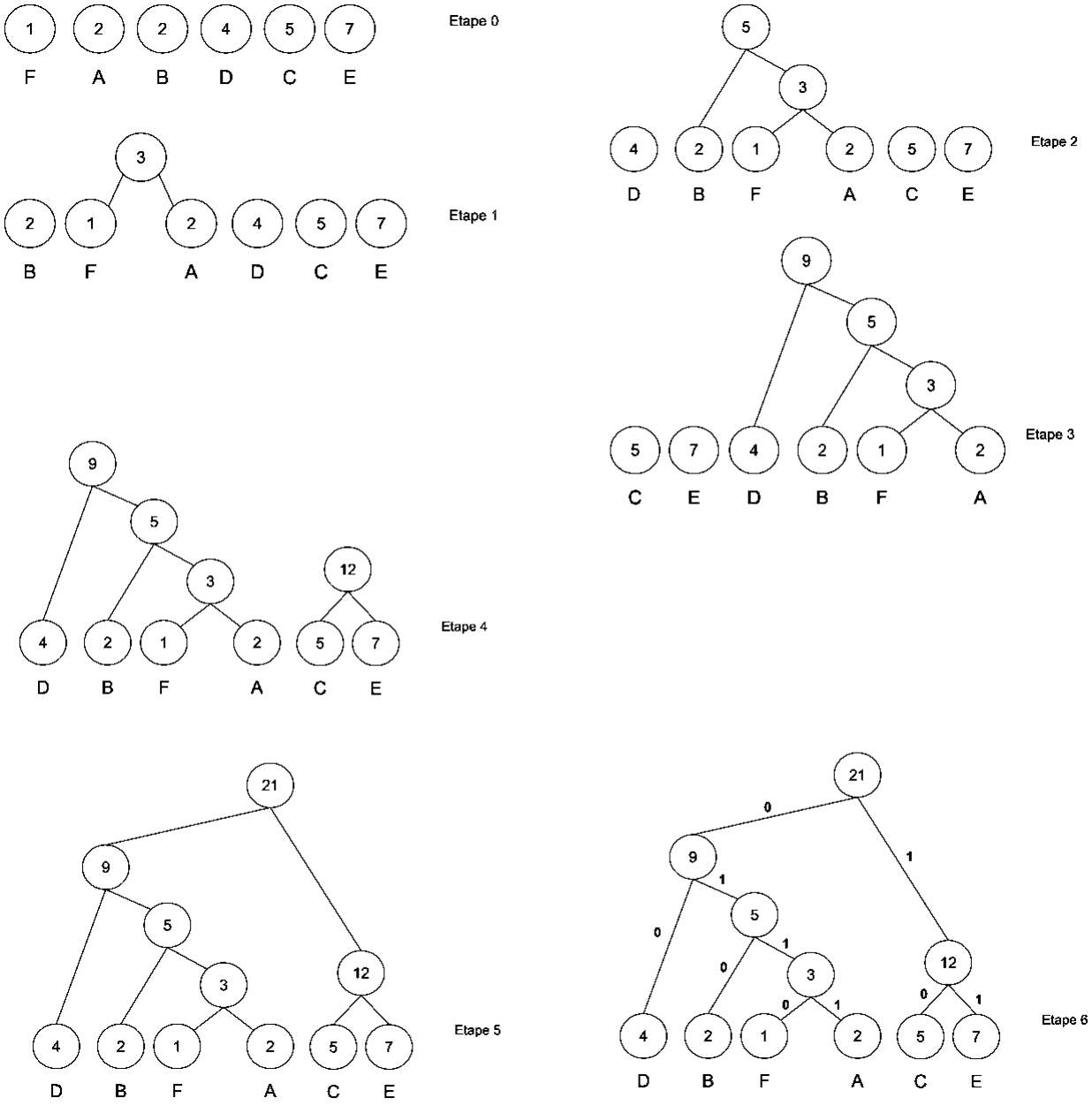


Figure 2. Construction de l'arbre de Huffman

6. Construire l'arbre de Huffman associé à la chaîne de caractères `txt`.
7. Préciser à quoi correspond le poids de la racine de cet arbre.

## Codage et compression à l’aide de l’arbre de Huffman

À l’aide de l’arbre de Huffman, on peut créer une table de codage où chaque symbole est codé par les bits lus sur le chemin entre la racine de l’arbre et la feuille correspondant au symbole. Dans l’exemple ci-dessus, la lettre ‘F’ serait codée par 0110 et la lettre ‘E’ par 11.

8. Indiquer le type de parcours à utiliser sur l’arbre de Huffman pour réaliser cette table de codage.

9. Donner la table de codage pour la chaîne de caractères `txt`.

10. Justifier le fait que le code de Huffman est un code de longueur variable.

Le codage du texte se fait ensuite caractère par caractère en utilisant la table de codage.

11. Coder la chaîne de caractères `txt` à l’aide du code de Huffman et de l’arbre construit à la question 6.

12. En reprenant le résultat déterminé dans la partie A, en déduire le taux de compression en % pour la variable `txt` et vérifier l’assertion du texte d’introduction : “On observe ainsi des réductions de taille de l’ordre de 20 % à 90 %.”

Le taux de compression est le ratio $\frac{\text{encombrement initial - encombrement final}}{\text{encombrement initial}}$.