Codes de Huffman : exercices

1. Supposons que le texte à compresser soit formé des lettres suivantes avec comme fréquence :

A	B	C	D	E	F	G	H	I	J
12	7	9	10	15	11	2	5	6	1

Construisez sur papier l'arbre de Huffman pour ces fréquences.

Codez ensuite le texte : `BADGE`

Décodez enfin la suite de bits 1111110011010100

2. Implémentez l'algorithme de Huffman pour pouvoir compresser et décompresser le livre « Alice au pays des merveilles »

Dans cet exercice, on vous demande d'implémenter une version très légèrement simplifiée de l'algorithme de Huffman à partir d'un squelette de code disponible sur moodle (les classes Huffman, HuffmanReadFile, HuffmanWriteFile). Comme cet algorithme n'est pas si facile, on va écrire l'algorithme par étape.

a. Implémentez la méthode computeFreq de la classe Huffman qui calcule la map des fréquences. Testez cette méthode avec la méthode main suivante (on testera avec le livre complet lorsque tout l'algorithme sera construit).

```
public static void main(String[] args) {
	String s="Bonjour! Au revoir!";
	Map<Character, Integer> freq = computeFreq(s);
	System.out.println(freq);}
```

Le résultat attendu est le suivant :

```
{ =2, !=2, A=1, B=1, r=3, u=2, e=1, v=1, i=1, j=1, n=1, o=3}
```

En effet, le caractère blanc apparait deux fois dans la String, le ' !' apparait deux fois également, …

b. Implémentez la méthode buildTree qui construit l'arbre de Huffman. Cette méthode prend en paramètre la Map des fréquences construites précédemment. L'arbre à construire utilisera la classe interne Node. Utilisez la théorie pour vous aider.
Vous pouvez tester partiellement la méthode avec la méthode main suivante :

```
public static void main(String[] args) {
	String s="Bonjour! Au revoir!";
	Map<Character, Integer> freq = computeFreq(s);
	Node root = buildTree(freq);
	System.out.println("Nbre de lettre dans la chaine
				de caractère à encoder: "+ root.freq );
	System.out.println("Fréquence des lettres dans le sous-arbre
					de gauche: "+ root.left.freq );
	System.out.println("Fréquence des lettres dans le sous-arbre de
					droite: "+ root.right.freq );
	}
```

Le résultat attendu est le suivant :

```
Nbre de lettre dans la chaine de caractère à encoder: 19
Fréquence des lettres dans le sous-arbre de gauche: 8
Fréquence des lettres dans le sous-arbre de droite: 11
```

c. Implémentez la méthode buildCode qui construit la Map des codes. Les clés de cette Map sont les caractères de la chaine à encoder, les valeurs sont leur code associé. Testez cette méthode avec la classe Main suivante :

```
public static void main(String[] args) {
	String s="Bonjour! Au revoir!";
	Map<Character, Integer> freq = computeFreq(s);
	Node root = buildTree(freq);
	Map<Character, String> code= buildCode(root);
	System.out.println(code);
}
```

Le résultat attendu est le suivant :

```
{ =010, A=0000, !=001, B=0001, r=101, u=100, e=1100, v=0111, i=11010,
j=11011, n=0110, o=111}
```

Le code du caractère blanc est 010, celui du caractère A est 0000, ….

d. Implémentez la méthode compress qui encode la chaine de caractère prise en paramètre en une chaine de bit 0 et 1 en utilisant la map des codes. B vaut 0001, o 111, n 0110, … -> Bonjour sera encodé par 000111101101101111110010. Testez cette méthode avec la classe Main suivante :

```
public static void main(String[] args) {
	String s="Bonjour! Au revoir!";
	Map<Character, Integer> freq = computeFreq(s);
	Node root = buildTree(freq);
	Map<Character, String> code= buildCode(root);
	String compress = compress(s, code);
	System.out.println(compress);
}
```

Le résultat attendu est le suivant :

```
000111101101101111110010100101000001000101011100011111111010101001
```

e. Implémentez la méthode expand qui à partir de la chaine codée (contenant des 0 et des 1) retrouve la chaine originale. Elle utilise pour cela l'arbre de Huffman donné en paramètre. Testez cette méthode avec la classe Main suivante :

```
public static void main(String[] args) {
        String s="Bonjour! Au revoir!";
        Map<Character, Integer> freq = computeFreq(s);
        Node root = buildTree(freq);
        Map<Character, String> code= buildCode(root);
        String compress = compress(s, code);
        String texteOriginal = expand(root,compress);
        System.out.println(texteOriginal);}
```

Le résultat attendu est bien évidemment : `Bonjour! Au revoir!`

f. Il est maintenant temps de tester avec le fichier 11-0.txt contenant le livre « Alice au pays des Merveilles ». On va utiliser des méthodes des classes HuffmanReadFile et HuffmanWriteFile qui permettent d'écrire des bits dans des fichiers (et non pas des String). Utilisez la méthode Main suivante :

```
public static void main(String[] args) throws IOException {
        String s=HuffmanReadFile.loadFile(new File("11-0.txt"));
        Map<Character, Integer> freq = computeFreq(s);
        Node root = buildTree(freq);
        Map<Character, String> code= buildCode(root);
        String compress = compress(s, code);
        HuffmanWriteFile.write("fichier_compresse",compress);
        String texteOriginal =
                expand(root,HuffmanReadFile.read("fichier_compresse"));
        System.out.println(texteOriginal);
}
```

L'algorithme peut prendre un peu de temps.

a) Vérifiez que le texte original est bien affiché à la console. Il ne peut y avoir de caractères supplémentaires comme « ee ». En présence de caractères supplémentaires, il faut repenser la méthode expand. La présence de ces caractères est due au stockage des informations sur le disque. En effet, les informations sont stockées sur le disque par bloc de 8 bits. Si la chaine de bit à la sortie de compress n'est pas un multiple de 8, le système va rajouter des bits supplémentaires. Ces bits ne doivent pas être décodés.

b) Améliorez votre algorithme pour qu'il s'exécute en moins d'une seconde. Si votre algorithme est lent, c'est surement dû aux nombreuses concaténations de String dans les méthodes compress et expand.

La taille de fichier_compresse est de 97 Ko alors que le fichier original était de 169 Ko. Le rapport des tailles entre les deux fichiers est de 57%. Utilisez Winzip pour compresser le fichier. Comparez les deux compressions.