POLYTECH

# COMPRESSION PAR CODAGE DE HUFFMAN

By William Burillon – IDU

# Sommaire



Lien du Github :
https://github.com/WilliamBurillon/CompressionByHuffman

# I. Codage et compression

## A. Codage

Liste des classes concernant le codage :

- CodageHuffman : Algorithme principale
- Codage : code en binaire suivant un dictionnaire
- CompressFile : Compresse le fichier en ASCII
- Nœud : Structure de Donnée
- HuffmanForest : Création d'un arbre suivant principe de Huffman

Fichier en Entrée :

-mode Statique : le texte .txt (example.txt)
la liste .dat (example_Freq.dat)

-mode semi-adaptatif : le texte .txt (example.txt)

Fichier en sortie :

« *_dico.dat » / « *_Huffcode.dat » / « *_Freq.dat »

# I. Codage et compression

## A. Codage

### Processus de codage :

1) Structuration des données

- Lecture du fichier texte → ArrayList<String[]> avec caractère + fréquence

```
LectureData reader2 = new LectureData(this.fileName + ".txt");
ArrayList<String[]> test =reader2.dicoText();
```

- Création des noeuds→ ArrayList<Noeud> +Création ArrayList<Nœud> vide

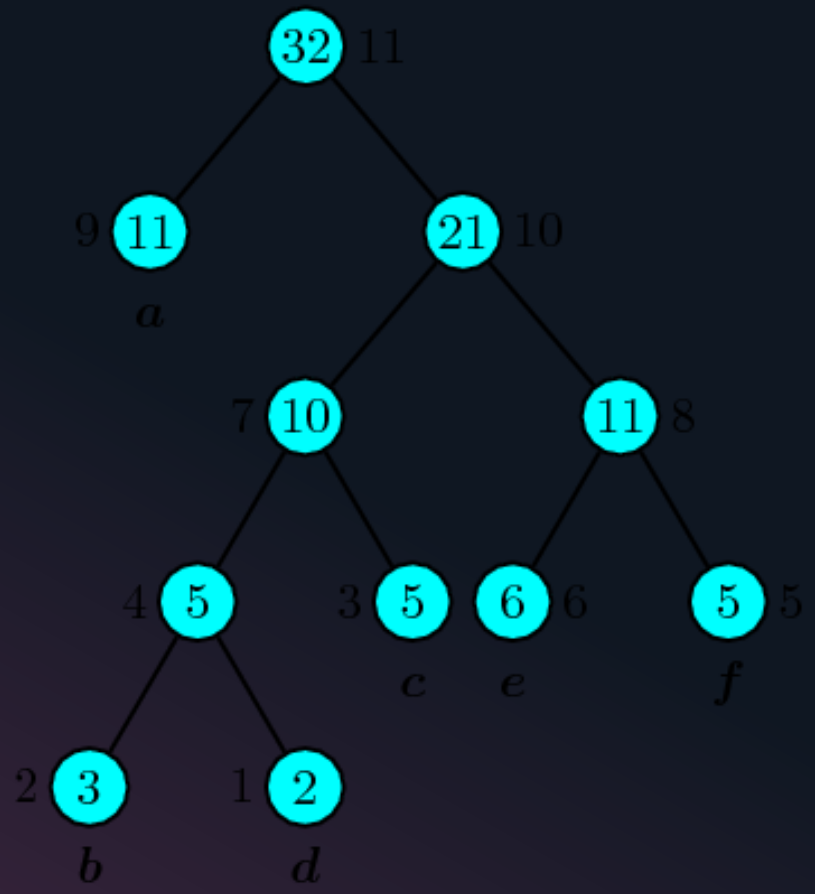
2) Création de des arbres ainsi que détermination du code binaire (Huffman)

- Pour tous les lettres présentes dans le textes (et leurs pondérations) → Copie par valeur de la liste de Nœud + Création d'un arbre d'Huffman + parcourt en profondeur pour trouver le code relatif au caractère courante et écriture dans un fichier

```
// make a Deep path of this tree to find the binary code related to the current character
// and this code will be write in the dictionary file
newDat.writeInFile(test.get(i)[0], hF.getForet().get(0).deepPath("", test.get(i)[0]));
```

# I. Codage et compression

## A. Codage




# Details de la création de l'arbre

### Création de l'arbre avec l'algorithme d'Huffman

- Au départ, on a une foret d'arbre composé que d'une seule feuille (lettre + fréquence d'apparition)
- On cherche les nœuds telle qu'ils aient un poids minimal
- On créer un nœud père, sans lettre, dont le poids et la somme des poids des deux fils
- On répète l'opération jusqu'à qu'il n'y ai plus qu'un arbre dans la foret
- Les arcs allant dans la direction du fils gauche sont pondérés à 0, ceux du fils droit sont pondérés à 1

# I. Codage et compression

## A. Codage


# Details de la détermination du code

« Toto et Tutu »

### Prenons l'exemple ci dessus

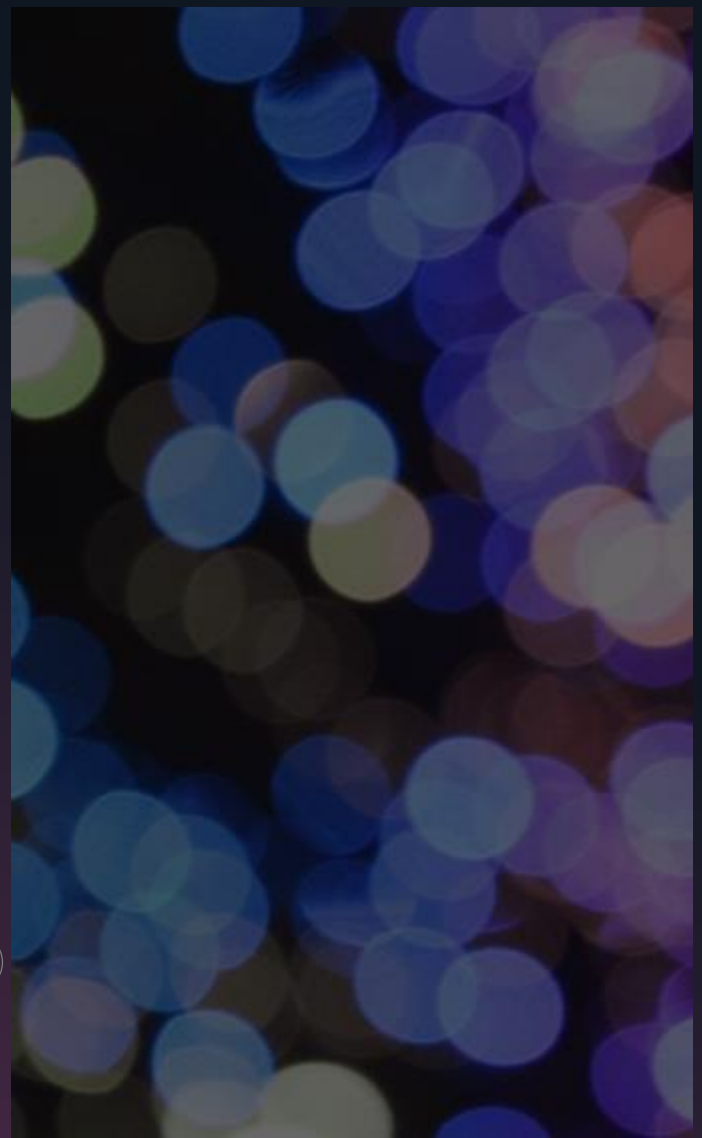
- Pour cette arbre représentant le texte ci-dessus, nous partons de la racine et effectué un parcours en profondeur jusqu'à arriver à lettre voulu
- Dans mon algorithme, à chaque parcours d'un nœud, la valeur du nœud prend -1 (pour éviter les boucles infinies)
- Pour E : le code binaire sera 000
- Pour U : le curseur parcourant l'arbre va s'arrêter à E, voir que c'est une feuille et revenir au nœud père en retranchant 0 au résultat, puis il va aller au nœud droit →

Le code de U est 001

Remarque : étant donné que je modifie la valeur des nœuds lors du parcours, j'ai du recrée un arbre pour chaque caractère, copier les nœuds par valeur, et non par adresse.

# I. Codage et compression

## A. Codage



### Suite de l'algorithme de codage

- Grace à la classe Codage, on lit le fichier texte original ainsi que le dictionnaire
- Pour chaque caractère, on regarde sa valeur binaire de dictionnaire et on ecrit dans le fichier "*_huffcode.dat"

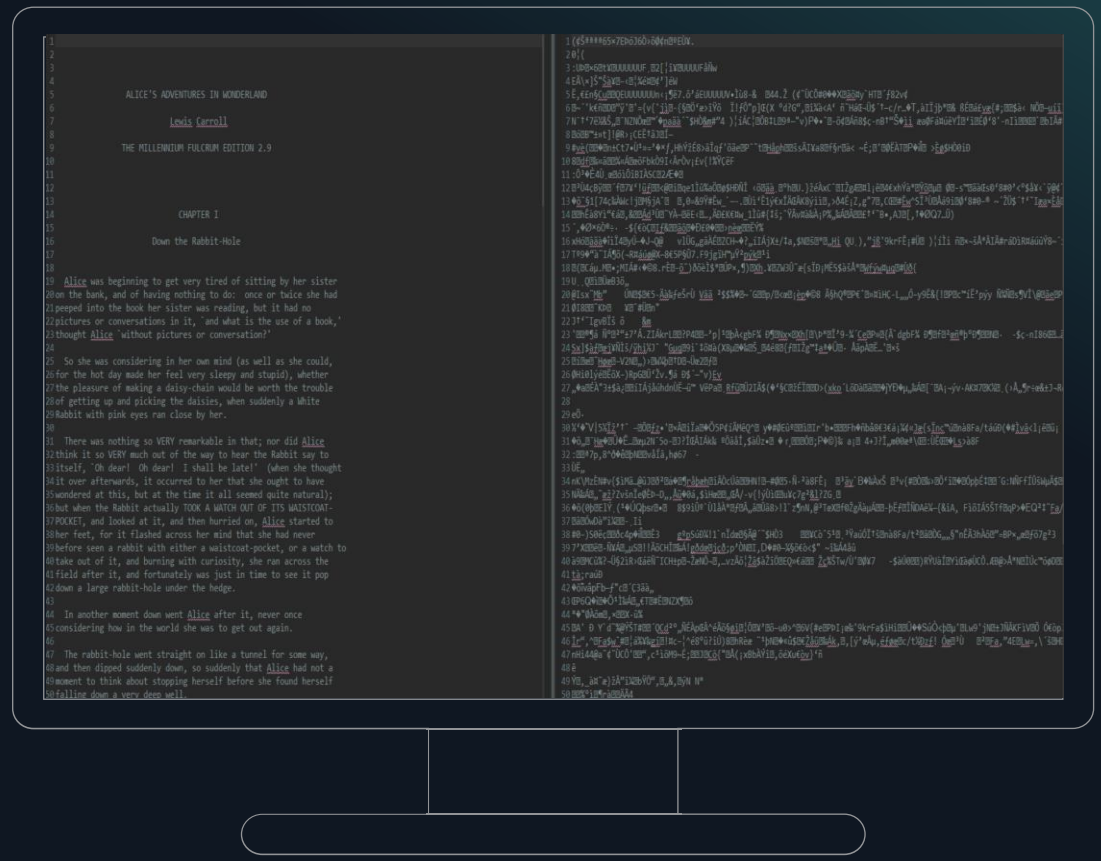
```
while ((line = br1.readLine())!=null) {

    for (int i =0; i < line.length();i++) {
    //for each line, getting char by char, cast it into a String and
    //research is binary code into the dictionary
        String car = String.valueOf(line.charAt(i));
        for (int j = 0; j< dico.size();j++) {
            // if the car is find in the dictionary, write the it binary code
            if (dico.get(j)[0].equals( car)){
                writer.write(dico.get(j)[1]);
            }
```

# I. Codage et compression

## B. Compression

- Lecture du fichier binaire « *_Huffcode,dat »
- Pour chaque octet, on utilise la méthode Integer.*parseInt(byte,2)* afin de convertir cet octet en integer, puis de la caster en char
- Pour la fin du fichier, si les bits ne représentent pas un octet, on rajoute des 0
- Ensuite on l'ecrit dans le fichier « *_compressed.txt » au format ASCII ISO 8859-1

## Resultat



# II. Décompression et décodage

## A. Décompression

Liste des classes concernant le décodage :

- DecodageHuffman : permet de décoder le texte
- Unzip : permet de décompresser le texte

Fichier en Entrée :

-mode semi-adaptatif : « *_compressed.txt »
« *_dico.dat »

Fichier en sortie :

String représentant le code binaire / « decodedtext.txt »

# II. Décompression et décodage

## A. Décompression

```
while ((line = br.readLine()) != null) {
    //For each line
    for (int i=0;i<line.length();i++ ) {
        //gettin the char by char
        char leChar = line.charAt(i);
        // cast it into a String which represents the binary code of the char
        String leRes = Integer.toBinaryString(leChar);

        // if the res is not a byte, so complete it with 0 at the begining
        if (leRes.length()<8) {
        String res3=leRes;
        while (res3.length()<8) {
            res3 = "0" + res3;
        }
        res = res+ res3;
        }

        else {
            res = res+leRes;
        }
        }
```

- Lecture du fichier « *_compressed.txt »
- Pour chaque caractère, utilisation de Integer.*toBinaryString(leChar)* -> retourne le code binaire du char, si taille < 8, ajout de “0” au début afin de le completer
- ajout de cet octet au String resultat

# II. Décompression et décodage

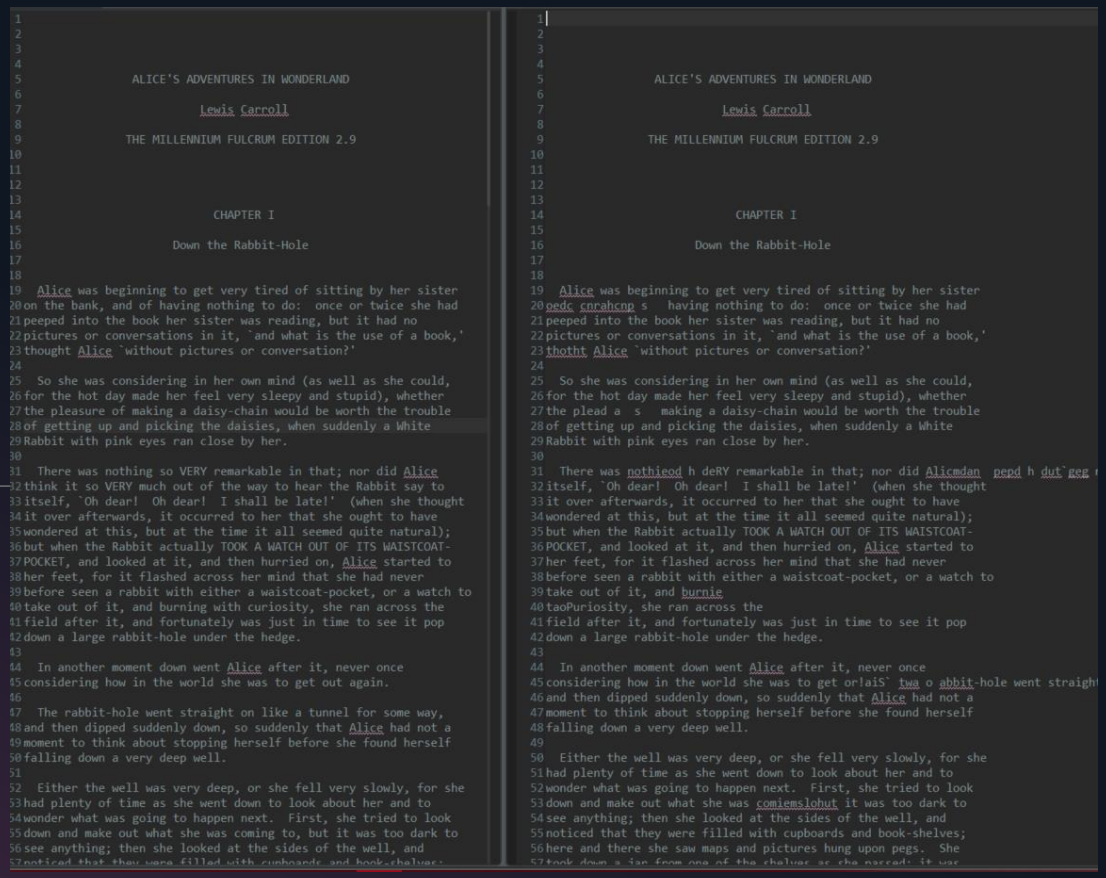
## B. Décodage

- Lecture du string retourner par l'algorithme décompression et du fichier « *_dico.dat »
- on lit bit par bit et ajout dans un String : test a chaque ajout s'il est présent dans le dictionnaire -> si trouvé, on prend le caractère associé et ajout dans le fichier « decoded.txt »

Remarque : on peut utilisé cette méthode étant donné que le préfixe du codage binaire est unique

```
LectureData lectDic = new LectureData(this.leDico);
ArrayList<String[]> dicToArray = lectDic.arrayListValue2();
try {
    BufferedWriter writer = new BufferedWriter(new FileWriter(new File("decodetext.txt"),true));

    String line= leFichierBinaire;
    System.out.println(line);
//
        String car = "";
        // read the String bit by bit and add it to car
        for (int i =0; i < line.length();i++) {
             car = car +  String.valueOf(line.charAt(i));
            // if the bit code is in the dictionary, so get the char related to it
            for (int j = 0 ; j<dicToArray.size();j++) {
                if (car.equals(dicToArray.get(j)[1])){
                    writer.write(dicToArray.get(j)[0]);
                    car = "";

                }

                }
            }

        if (line!=null) {
            writer.write("\n");
        }
```

# III. Résultats et Analyse



ALICE'S ADVENTURES IN WONDERLAND

Lewis Carroll

THE MILLENNIUM FULCRUM EDITION 2.9

CHAPTER I

Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, `and what is the use of a book,' thought Alice `without pictures or conversation?'

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, `Oh dear! Oh dear! I shall be late!' (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and

ALICE'S ADVENTURES IN WONDERLAND

Lewis Carroll

THE MILLENNIUM FULCRUM EDITION 2.9

CHAPTER I

Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister
oedc cnrahcnp s having nothing to do: once or twice she had
peeped into the book her sister was reading, but it had no
pictures or conversations in it, `and what is the use of a book,'
thotht Alice `without pictures or conversation?'

So she was considering in her own mind (as well as she could,
for the hot day made her feel very sleepy and stupid), whether
the plead a s making a daisy-chain would be worth the trouble
of getting up and picking the daisies, when suddenly a White
Rabbit with pink eyes ran close by her.

There was nothieod h deRY remarkable in that; nor did Alicmdan pepd h dut`geg
itself, `Oh dear! Oh dear! I shall be late!' (when she thought
it over afterwards, it occurred to her that she ought to have
wondered at this, but at the time it all seemed quite natural);
but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-
POCKET, and looked at it, and then hurried on, Alice started to
her feet, for it flashed across her mind that she had never
before seen a rabbit with either a waistcoat-pocket, or a watch to
take out of it, and burnie
taoPuriosity, she ran across the
field after it, and fortunately was just in time to see it pop
down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once
considering how in the world she was to get or!aiS` twa o abbit-hole went straigh
and then dipped suddenly down, so suddenly that Alice had not a
moment to think about stopping herself before she found herself
falling down a very deep well.

Either the well was very deep, or she fell very slowly, for she
had plenty of time as she went down to look about her and to
wonder what was going to happen next. First, she tried to look
down and make out what she was comiemslohut it was too dark to
see anything; then she looked at the sides of the well, and
noticed that they were filled with cupboards and book-shelves;
here and there she saw maps and pictures hung upon pegs. She

Exemple avec extrait_alice.txt

## Résultat Avant / Apres Compression

| Fichier | Taille Fichier Orgine | Taille Fichier Compressé |
|---|---|---|
| Montexte.txt | 36 o | 18 o |
| Extrait_alice.txt | 11,8 Ko | 6,49 Ko |
| Alice29.txt | 148 Ko | 82,5Ko |

Remarque :

- on remarque que le fichier à réduit de 50% à 40% suivant ça taille d'origine
- Taux de ressemblance estimé : 85% à 94% de taux de ressemblance suivant les textes

Causes :

Problème de décompression, le code binaire en entrée n'est pas e même que le code binaire en sortie ( Octet rajouté lors de la décompression

# IV. Limites et améliorations possibles

## Limites de l'encodage :

- Création de l'arbre : ajouter une contrainte sur la sélection des nœuds si même fréquence -> permet de mieux contrôler
- Optimisation de la place prise en mémoire lors de la recherche du code binaire des caractères

## Limites du décodage:

- On prends un alphabet (correspondance charactère – code binaire) pour le décodage -> préférable de prendre un fichier contenant les fréquences et créer un arbre de décodage.