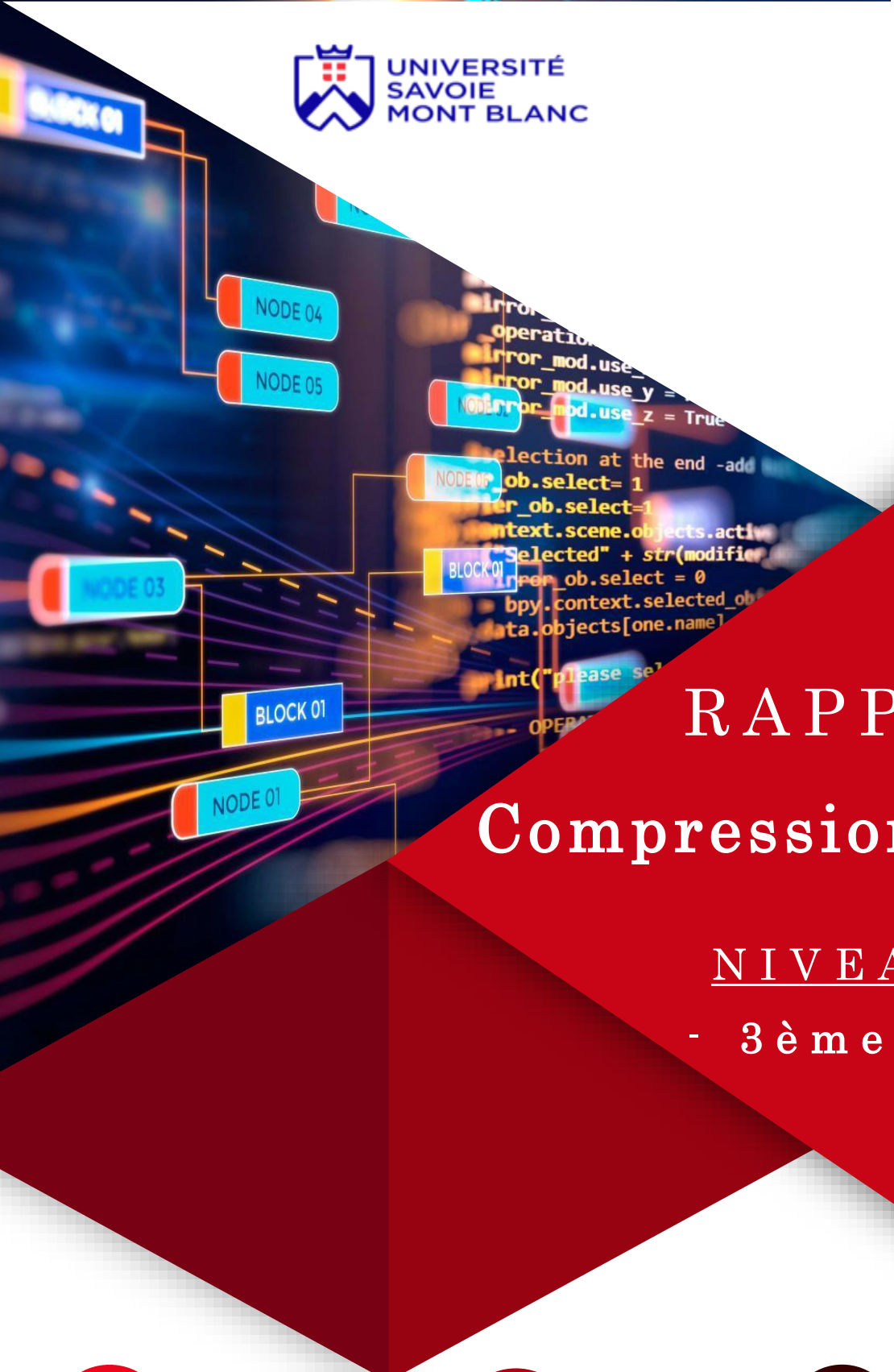
UNIVERSITÉ SAVOIE MONT BLANC

POLYTECH ANNECY-CHAMBÉRY



RAPPORT

Compression Huffman

NIVEAU :

- 3ème Année IDU

ENCADRE PAR :

- Mr. DAHHANI Abdelhafid
- Mme. CIMPAN Sorana
- Mme. ALLAOUI Ilham

DATE :

- LE 30/05/2022

REALISE PAR :

- BOUMAHREZ Ouijdane

Résumé

Mon projet portera sur compression de données par codage de Huffman tout en proposant une structure de données permettant de modéliser ce dernier.

Tout d'abord, je présenterai mon projet, tout en définissant les consignes qu'il faut suivre et les objectifs du projet. Ensuite, pour mieux comprendre ma gestion organisationnelle, je me pencherai plus précisément sur les méthodes et les outils d'organisation que j'ai utilisé pour gérer ce projet. En ce qui concerne la gestion technique, je présenterai plus en détails, la décomposition fonctionnelle de mon projet, les aspects techniques ainsi que le mode d'emploi de mon programme. En guise de conclusion, je donnerai le bilan de mon projet tout en présentant : les objectifs réalisés, les taches qui me manquent à faire et les lacunes rencontrées.

Abstract

My project will focus on data compression by Huffman coding while proposing a data structure to model it.

First, I will present my project, while defining the instructions to be followed and the objectives of the project. Then, to better understand my organizational management, I will look more specifically at the organizational methods and tools that I used to manage this project. As far as technical management is concerned, I will present in more detail the functional breakdown of my project, the technical aspects as well as the instructions for use of my program. By way of conclusion, I will give the results of my project while presenting: the objectives achieved, the tasks that I lack to do, and the shortcomings encountered.

Table de matières

Introduction générale

Dans le cadre de notre cycle d'ingénieur, j'ai suivi un enseignement qui m'a permis d'avoir des bases en gestion de projet, et la gestion des structures de données.

Dans un travail individuel, il m'a été proposé un projet « Compression de données par codage de Huffman ». Ce projet a pour but de mettre en application tout ce que nous avions eu à apprendre, et mettre en œuvre nous capacités organisationnelles. Comme il est une mise en application des structures de données et algorithmes présentés dans le module « Graphes et Langages ». Il prend la forme de développements logiciels dans le respect des méthodes de gestion de projets informatiques vues dans le module « Gestion de projets ». En particulier, des outils appropriés à un développement modulaire des programmes et à une gestion de l'évolution du code produit seront exploités.

Ce projet concerne la version semi-adaptative de l'algorithme dans laquelle le texte à coder est tout d'abord lu intégralement de façon à construire l'alphabet et déterminer les fréquences d'apparition des éléments de l'alphabet.

Mon programme devra réaliser la phase de codage d'un texte fourni selon les trois étapes suivantes :

- Détermination de l'alphabet et des fréquences de caractères

- Construction de l'arbre de codage

- Codage et compression du texte initial

Puis déterminer et afficher :

- le taux de compression obtenu

- le nombre moyen de bits de stockage d'un caractère dans le texte codé

Pour le résultat qu'il faut fournir à la fin, il faut que pour chacun des textes fournis (<nom>.txt), votre programme devra générer un fichier du texte compressé (<nom>_comp.bin) et un fichier de description de l'alphabet utilisé avec les fréquences de caractère (<nom>_freq.txt). Ce dernier devra contenir sur une première ligne la taille de l'alphabet (nombre de caractères) puis pour chacun d'entre eux le caractère suivi de sa fréquence. Les caractères de l'alphabet seront rangés par fréquence croissante puis par valeur de code ASCII (ordre alphabétique).

Partie I : FONCTIONNEMENT HIFFMAN SEMI-ADAPTATIF

1. *Lecture des fichiers*
2. *Arbre binaire de recherche*
3. *Exécution en console*
4. *Définition des classes et des fonctions*
5. *Points d'amélioration*

1. *Lecture des fichiers :*

Avant de débuter la programmation, j'ai dû comprendre le principe.

Les premières séances ont donc été consacrées à explorer la documentation, ainsi que les fichiers fournis dans le dossier Données et comprendre comment j'y vais m'en servir.

Les fichiers disponibles et traités :

- Alice.txt
- extraitalice.txt
- textesimple.txt

Une fois la compréhension acquise, j'ai pu réaliser le programme permettant d'implémenter ma structure de données tout en utilisant des méthodes qui facilitent la saisie de données.

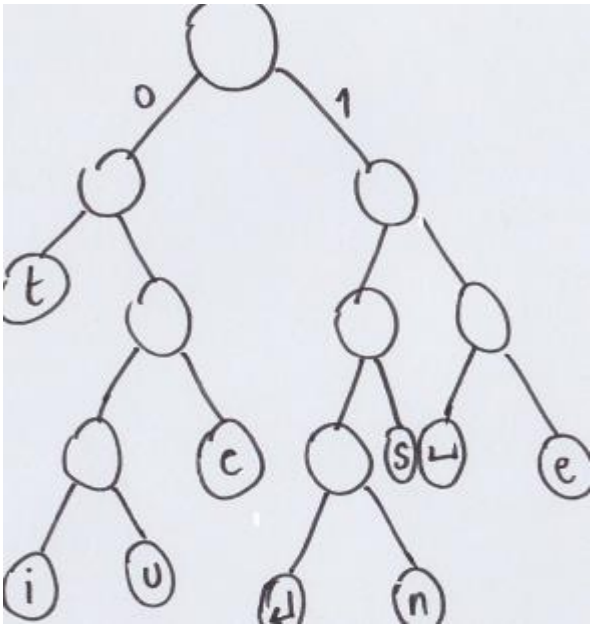
On sait déjà que Le codage de Huffman, du nom de son concepteur, est une méthode statistique de compression de données. Son principe est de remplacer un caractère (ou symbole) par une suite de bits de longueur variable. L'idée sous-jacente est de coder ce qui est fréquent sur peu de bits et au contraire ce qui est rare sur des séquences de bits plus longues. Il permet une compression sans perte, c'est-à-dire qu'une suite de bits strictement identique à l'originale est restituée par décompression. Il nécessite cependant que soit connues (ou estimées) les fréquences d'apparition des différents symboles à coder. Il existe ainsi plusieurs variantes de l'algorithme de Huffman (statique, semi-adaptatif ou adaptatif) aujourd'hui utilisées dans des algorithmes de compression de fichiers tels que gzip.

Ce sujet concerne la version semi-adaptative de l'algorithme dans laquelle le texte à coder est tout d'abord lu intégralement de façon à construire l'alphabet et déterminer les fréquences d'apparition des éléments de l'alphabet.

2. *ARBRE BINAIRE DE RECHERCHE :*

Le programme est sous forme d'une arborescence dont le nœud racine est le départ.

Création d'un arbre binaire de recherche basée sur la fréquence d'apparition des caractères : plus un caractère est présent, plus sa place dans l'arbre est haute (les nœuds représentant les caractères sont obligatoirement des feuilles).

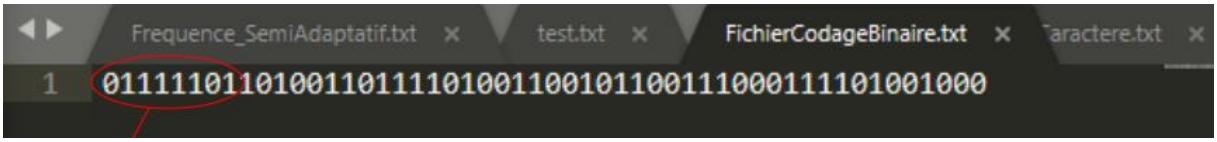


On lit l'arbre depuis sa racine : quand on va vers un fils gauche, on ajoute un « 0 » (zéro) à la valeur littérale binaire du caractère, et un « 1 » quand c'est un fils droit. On attribue alors à chaque caractère sa valeur binaire.

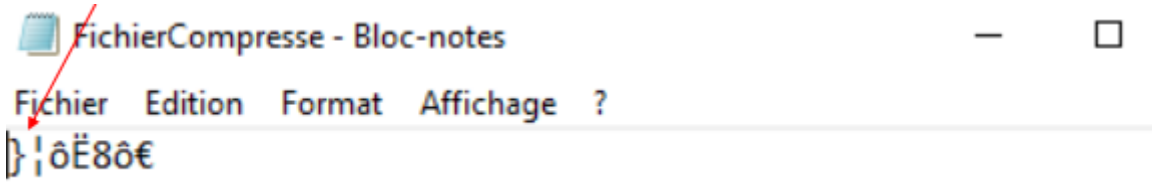
Frequence_SemiAdaptatif.txt | FichierCodageParCaractere.txt

```
1
2   1000
3    110
4  c 011
5  e 111
6  i 0100
7  n 1001
8  s 101
9  t 00
10 u 0101
```

On parcourt une seconde fois le texte : pour chaque caractère, on ajoute sur un fichier texte sa valeur binaire.

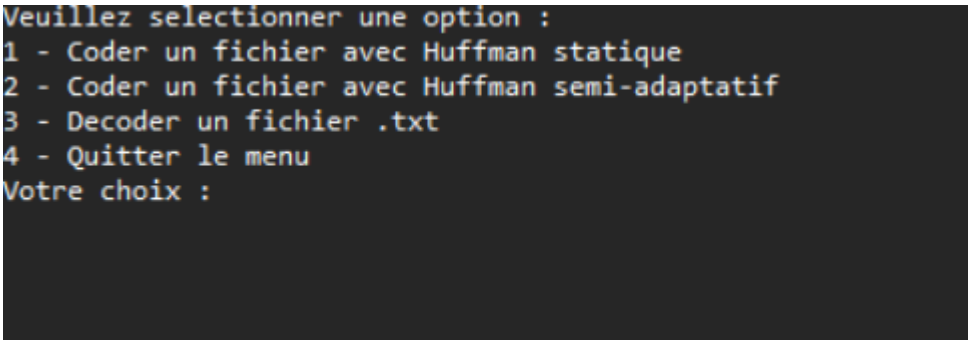


On encode chaque octet (8 bits) par le caractère qui correspond cette chaine de 8 bits et l'ajoute dans le fichier résultat.



3. *EXECUTION DE LA CONSOLE :*

Le mode d'emploi du programme se représente sous forme d'un menu à 4 options :



Déterminant plus en détails ces 4 options :

- **1er Choix : codage avec Huffman statique**

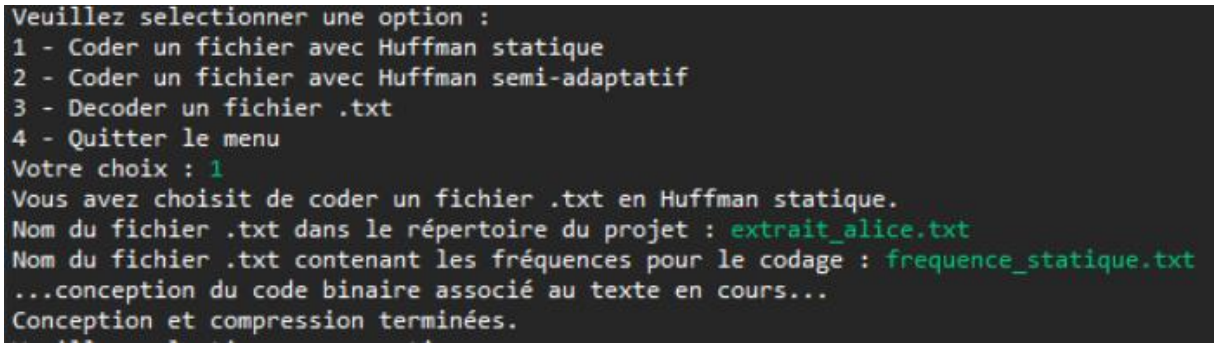
On rentre le nom du fichier texte à compresser et le fichier contenant les fréquences d'apparition de chaque caractère.

On retrouvera dans le dossier du projet 3 nouveaux fichiers :

• FichierCodageParCaractere.txt : codage binaire de chaque caractère

• FichierCodageBinaire.txt : fichier de 0 et de 1 représentant le fichier à compresser

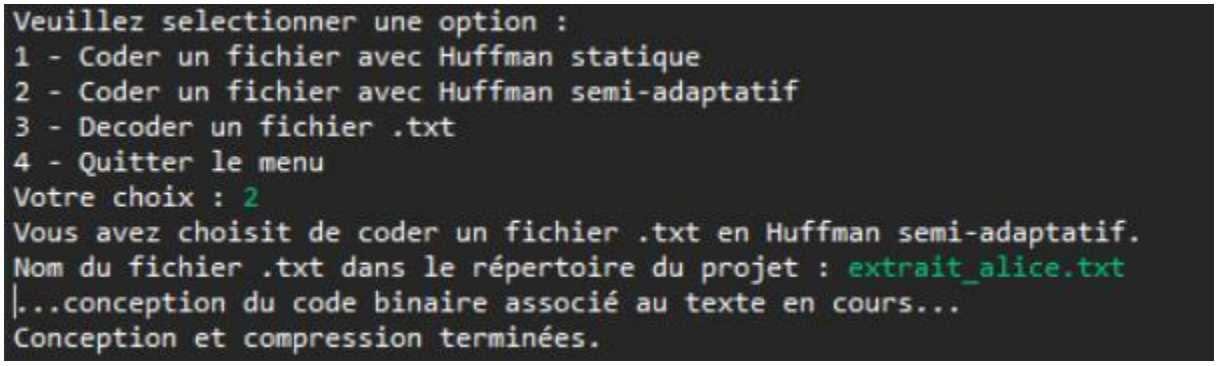
• FichierCompresse.txt : fichier résultat de taille plus faible



- **2ème Choix : codage avec Huffman semi-adaptatif**

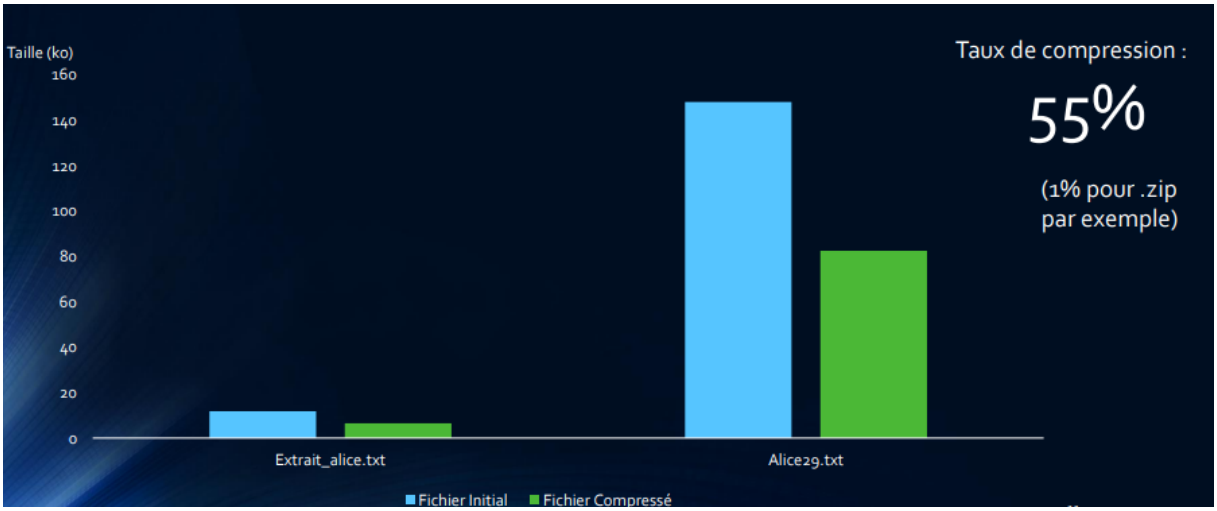
On rentre le nom du fichier texte à compresser et le fichier est codé avec les fréquences d'apparition de chaque caractère propre au fichier.

On retrouvera dans le dossier du projet 4 nouveaux fichiers, les trois cités précédemment ainsi que le fichier frequence_SemiAdaptatif.txt où l'on peut retrouver tous les caractères présents dans le texte ainsi que le nombre d'apparition.



Aperçu du fichier texte compressé :

Etude de la taille des fichiers à coder :



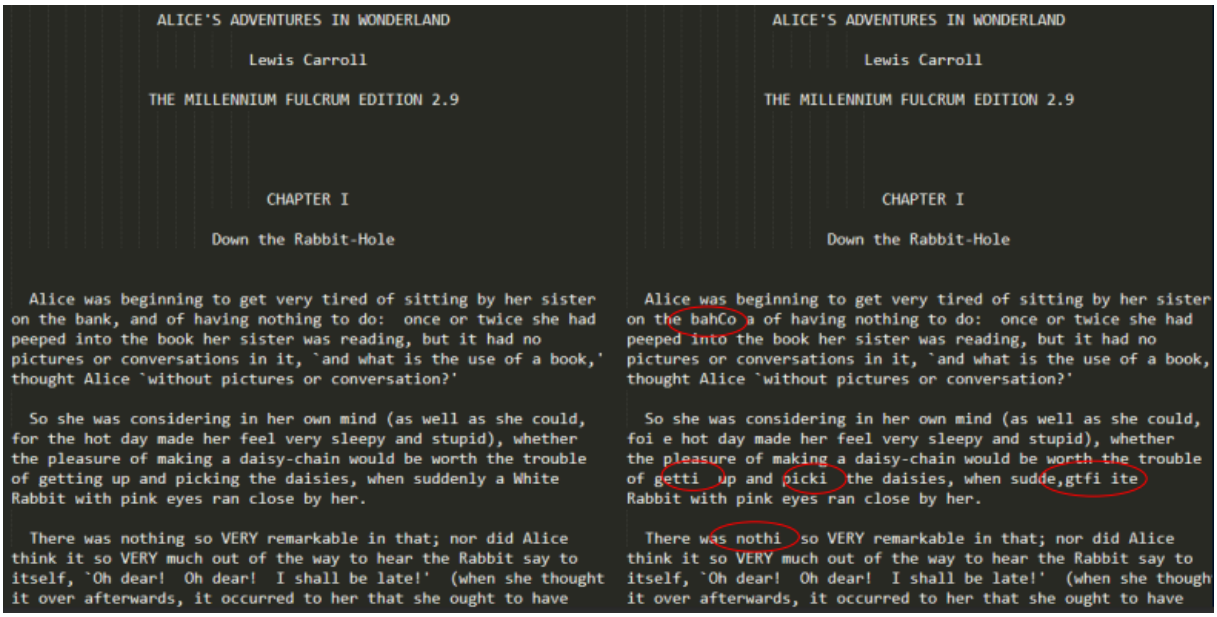
- **3ème Choix : décodage fichier encodé par Huffman avec valeur binaire de chaque caractère**

On rentre le nom du fichier texte à décompresser et le fichier est décodé avec le dictionnaire des valeurs binaires de chaque caractère.

On retrouvera dans le dossier du projet 2 nouveaux fichiers :

• DecodageBinaire.txt : contient des suites de 0 et 1 correspondant au décodage des caractères Encodés.

• FichierDecode.txt : fichier résultat contenant théoriquement le même texte qu'au départ.



Ici on a 52 différences dans 2203 mots, donc le taux est 98%.

4. *DEFINITION DES CLASSES ET DES FONCTIONS :*

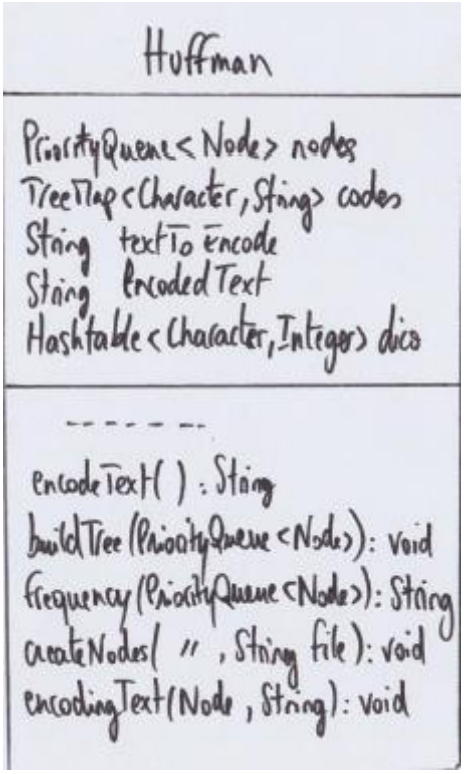
Ici je présenterai les classes et les fonctions que j'ai utilisées, alors dans mon programme on trouvera une classe main sur Java et 3 classes auxiliaires qui sont :

- Huffman
- Node
- ReadFileToCompress

➢ **MAIN :**

Une classe Main qui gère le menu et les différentes déclarations d'objets, ainsi que les appels aux méthodes définies dans les autres classes.

➢ **CLASSE HUFFMAN.JAVA :**




```java
import java.io.BufferedReader;

public class Huffman {

    private PriorityQueue<Node> nodes = new PriorityQueue<>((o1, o2) -> (o1.getValue() < o2.getValue()) ? -1 : 1);
    private TreeMap<Character, String> codes = new TreeMap<>();
    private String textToEncode = "";
    private String encodedText = "";
    private Hashtable<Character, Integer> dico = new Hashtable<Character, Integer>();


    public Huffman(String textToEncode){
        this.textToEncode = textToEncode;
    }

    public String getTextToEncode() {
        return this.textToEncode;
    }

    public PriorityQueue<Node> getNodes(){
        return this.nodes;
```

Les fonctions que j'ai utilisé dans cette classe sont les suivants :

- PriorityQueue : liste classée par ordre automatiquement (ici par ordre de fréquence d'apparition des caractères) -> utilisé pour la création de l'arbre.

- TreeMap : création d'un arbre binaire (fils gauche et droit) qui sera nécessaire pour la compression.

- frequency (…) : méthode qui retourne un String avec la fréquence d'apparition de chaque caractère (ce que l'on trouve dans le fichier frequence_SemiAdaptatif.txt, utilisé que dans l'option 2).

- createNodes (…) : une fois les fréquences acquises, on va créer les nœuds qui iront dans l'arbre binaire avec leur caractère et leur fréquence d'apparition (nbApparition/nbTotalCaractère) pour le moment. On les met dans la liste nodes.

- buildTree(…) : une fois tous les nœuds crées, cette méthode va supprimer petit à petit les feuilles les plus basses et mettre les nœuds dans les attributs fils du nœud père. On a finalement un nœud père avec des fils successifs (principe de l'arbre binaire).

- encodingText (…) : on parcourt tous les nœuds à partir du nœud racine, en retenant 0 pour les fils gauche et 1 pour les fils droits, et lorsqu'on atteint une feuille on enregistre sa valeur binaire dans l'un de ses attributs.

- encodeText() : retourne un String qui correspond à l'encodage binaire du texte de départ.

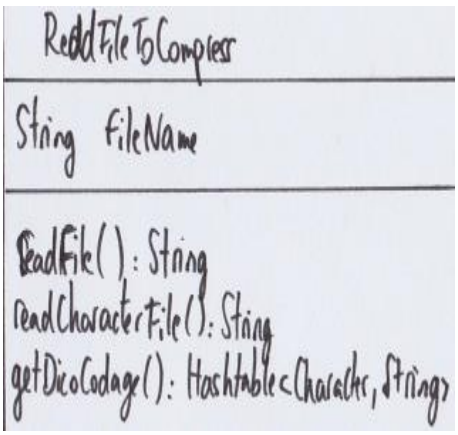
➢ **CLASSE ReadFileToCompress.java :**

Classe qui permet de lire différents fichiers pour le codage et pour le décodage, et possède différentes méthodes utilisées dans la classe Main :

- readFile() : retourne un String qui contient tout le fichier texte
-
- readCharacterFile() : retourne un String avec des 0 et 1 correspondant au décodage du fichier

Compressé

- getDicoCodage() : retourne un dictionnaire avec la valeur binaire de chaque caractère

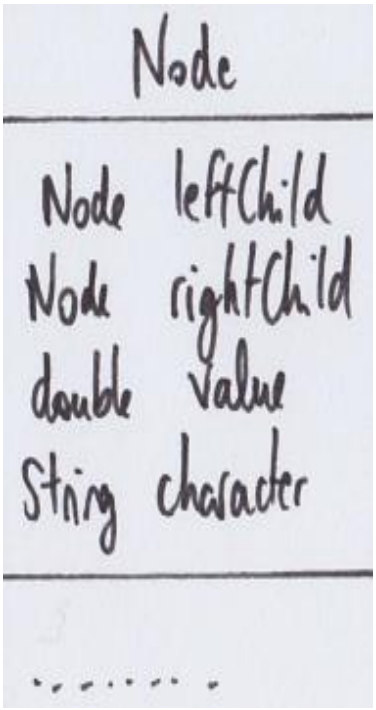



```java
public class ReadFileToCompress {

    private String fileName;

    public ReadFileToCompress(String fileName) {
        this.fileName = fileName;
    }
    public String readFile() throws IOException {
        String text = "";
        BufferedReader br = new BufferedReader(new FileReader(fileName));
        String line;
        while ((line = br.readLine()) != null) {
            text = text + line + "\n";
        }
        br.close();
        return text;
    }
```

- **CLASSE Node.java :**



Huffman.java | Main.java | Node.java | ReadFileToCompress.java

```java
public class Node {

    private Node leftChild;
    private Node rightChild;
    private double value;
    private String character;

    public Node(double value, String character) {

        this.value = value;
        this.character = character;
        this.leftChild = null;
        this.rightChild = null;
    }

    public Node(Node left, Node right) {
        this.value = left.value + right.value;
        character = left.character + right.character;
        if (left.value < right.value) {
            this.rightChild = right;
            this.leftChild = left;
        } else {
            this.rightChild = left;
            this.leftChild = right;
        }
    }
```

A l'aide de l'arbre que nous avons créé, nous allons maintenant donner à chaque caractère son code binaire. Nous parcourons donc l'arbre en profondeur et si nous partons dans un fils, nous ajoutons donc '0' au code binaire du fils et '1' à son fils droit jusqu'à ce que nous soyons à une feuille et que nous ayons parcourus tout l'arbre. Une fois que le nœud est une feuille, nous l'ajoutons à un dictionnaire pour lettre comme clé le caractère et comme valeur le code binaire du caractère.

5. *LES POINTS D'AMELIORATION :*

Le projet étant maintenant fini, je m'aperçois de certaines améliorations qui auraient pu être implémenté avec un peu plus de temps.

Tout d'abord le premier point d'amélioration aurait été de créer une sélection de fichiers à compresser.

Donc en faisant une entrée qui nous présente les fichiers ayant le bon format pour pouvoir être compressé.

Une seconde piste d'amélioration serait l'ajout d'une interface graphique. Tout d'abord pour présenter plus proprement la sélection de dossiers. Puis par la suite afficher un résumé du fichier fréquence ou bien une petite représentation de l'arbre binaire.

PARTIE II : GESTION DE PROJET

1. ORGANISATION ET TACHES

1. *ORGANISATION ET TACHES :*

A partir du cours, nous avions en notre possession tous les outils nécessaires au bon déroulement d'un projet. Etant dans une formation d'ingénieur, la gestion ainsi que le respect des délais est d'autant plus importante et essentielle dans la conception d'un projet.

La gestion de projet, permet de distribuer entre soi même les différentes tâches à réaliser, mais également avoir une base pour surveiller l'évolution du projet.

Dès le début du projet une liste de toutes les tâches à accomplir a été établi après avoir pris connaissance du sujet et des éléments importants.

Par la suite, j'ai essayé d'affecter des différentes tâches à faire dans chaque séance sur la programmation et d'autre sur ce qui concerne la gestion.

Je fais le point à chaque session et à prévoir et planifier les avancées pour la prochaine séance.

Afin d'avoir une réelle planification, je me suis servie du TRELLO qui m'a permis d'avoir une vision assez claire sur les dates de début et de fin de différentes tâches, ce qui m'a permis d'avoir conscience du temps restant.

Afin de compléter la partie organisationnelle, j'ai utilisé GitHub pour gérer la partie informatique.

Cette plateforme m'a permis, d'y mettre tous les fichiers en lien avec le programme, où tout y avait accès puisque c'était public.

Pour la mise en place du projet, il nous a fallu une interface pour la communication, nous avions donc utilisé Discord, pour se tenir au courant des avancées, pour le cas où l'un d'entre nous rencontrait des difficultés nous pouvions échanger, et réaffecter certaines tâches, il nous a aussi permis de faire un partage d'écran afin de voir en direct certains test du code. De plus, plusieurs salons ont été créé en fonction des différentes tâches.

BILAN GENERAL

Ce projet fut une véritable expérience enrichissante pour moi. Il m'a permis de forger des convictions sur ce que je sais et aime faire. Comme il m'a permis d'acquérir des compétences relationnelles organisationnelles en autonomie, ainsi que des compétences techniques.

Dans un premier temps, ce projet fut un terrain de jeu parfait pour développer mes soft-skills, entre gestion, organisation et surtout communication et recherche.

Parmi les difficultés rencontrées au cours de ce projet était au niveau technique. En effet, l'utilisation de GitHub, ou du Trello et surtout le codage sur java n'était pas intuitif pour moi. Je suis un à l'aise en ce qui concerne la gestion de projet et un peu moins à l'aise en technique. Cependant, ceci n'était pas un obstacle pour moi, en regroupant mes habiletés complémentaires et avec un peu de recherche sur internet (tuto Youtube …) et demander de l'aide à chaque fois que je me bloque, tout cela m'a permis d'accomplir mon projet plus rapidement et avec plus d'efficacité.

Références et liens annexes

Liens :

- https://trello.com/b/GD27bjXd/compression-huffman
- https://github.com/Ouij2902/Compression_de_donnees