

PROJ 631 – Projet Algorithmique

COMPRESSION DE DONNÉES PAR CODAGE DE HUFFMAN



Corentin BALLAZ – FI3 IDU

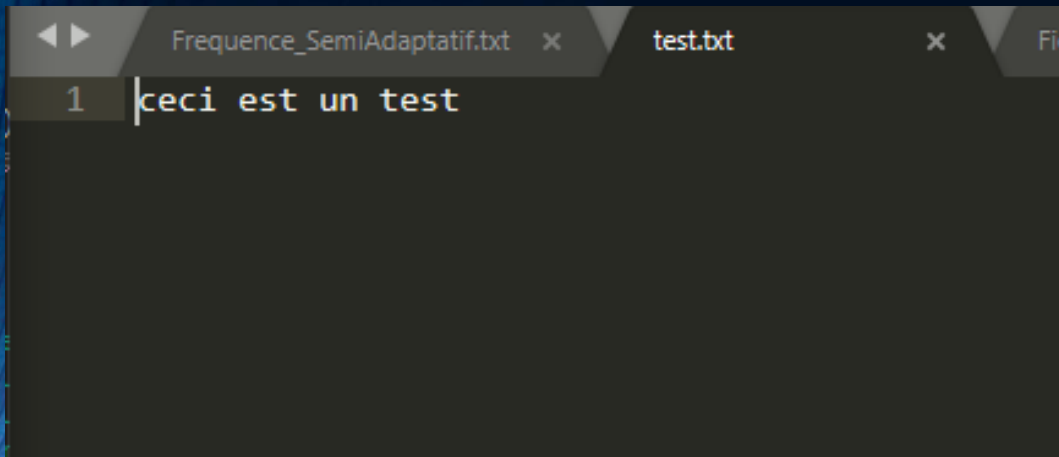
<https://github.com/CorentinBallaz/HuffmanCoding>

Rappel du projet et de ses objectifs :

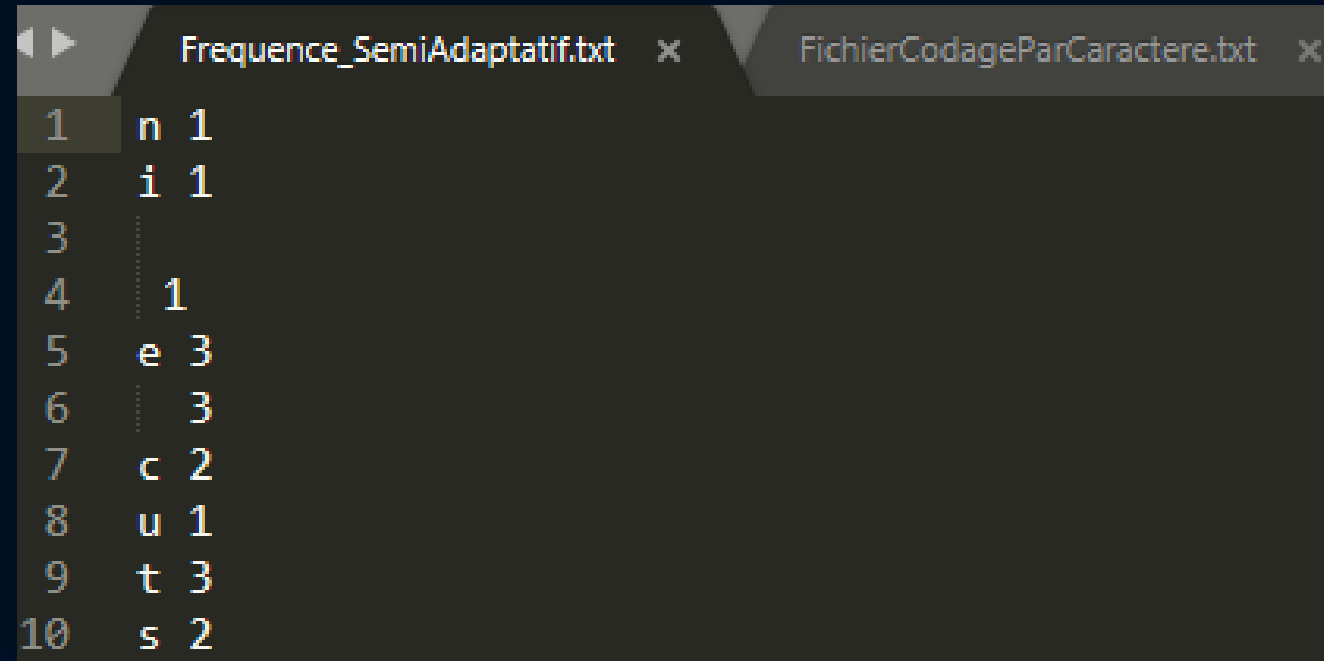
- Compression d'un fichier d'extension .txt à l'aide du codage de Huffman (programme Java)
- Création d'un fichier d'extension .txt de taille plus faible
- Fonctionnement à l'aide d'un codage par fréquence :
 - Statique pour un codage où l'on définit la fréquence d'apparition des caractères
 - Semi-adaptatif pour un codage où l'on utilise la fréquence d'apparition des caractères propre au texte
- Projet sur 3 séances (12h)
- Partie décodage en bonus

EXEMPLE DE FONCTIONNEMENT HUFFMAN SEMI-ADAPTATIF

1) Lecture d'un texte et création automatique d'un dictionnaire contenant le nombre d'apparition de chaque caractère.



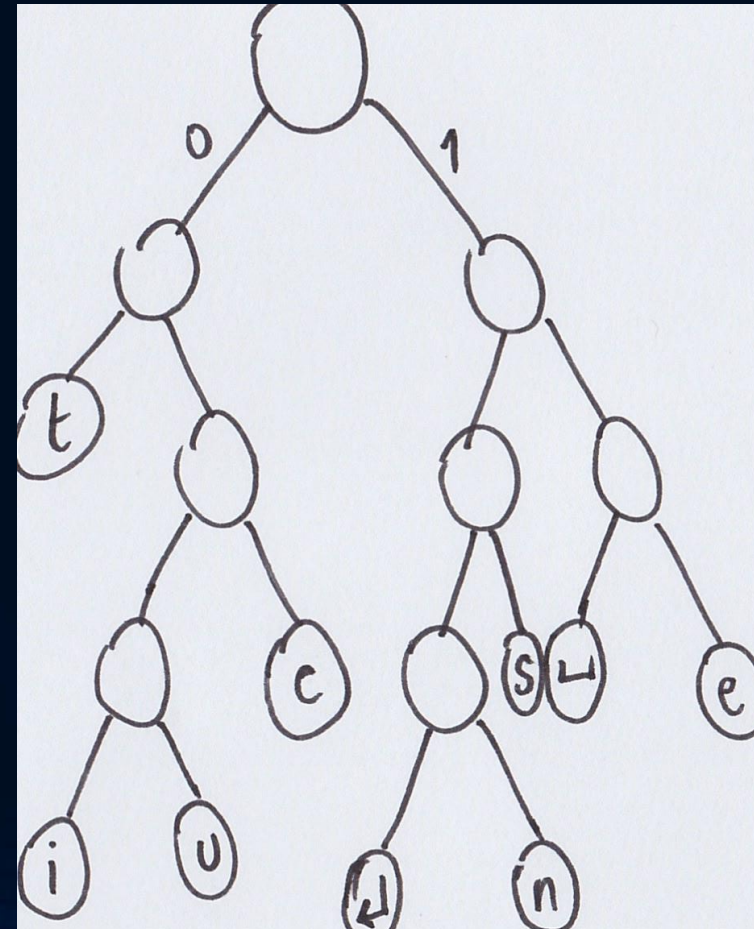
```
1 | ceci est un test
```



```
1 | n 1  
2 | i 1  
3 |  
4 | 1  
5 | e 3  
6 | 3  
7 | c 2  
8 | u 1  
9 | t 3  
10 | s 2
```


EXEMPLE DE FONCTIONNEMENT HUFFMAN SEMI-ADAPTATIF

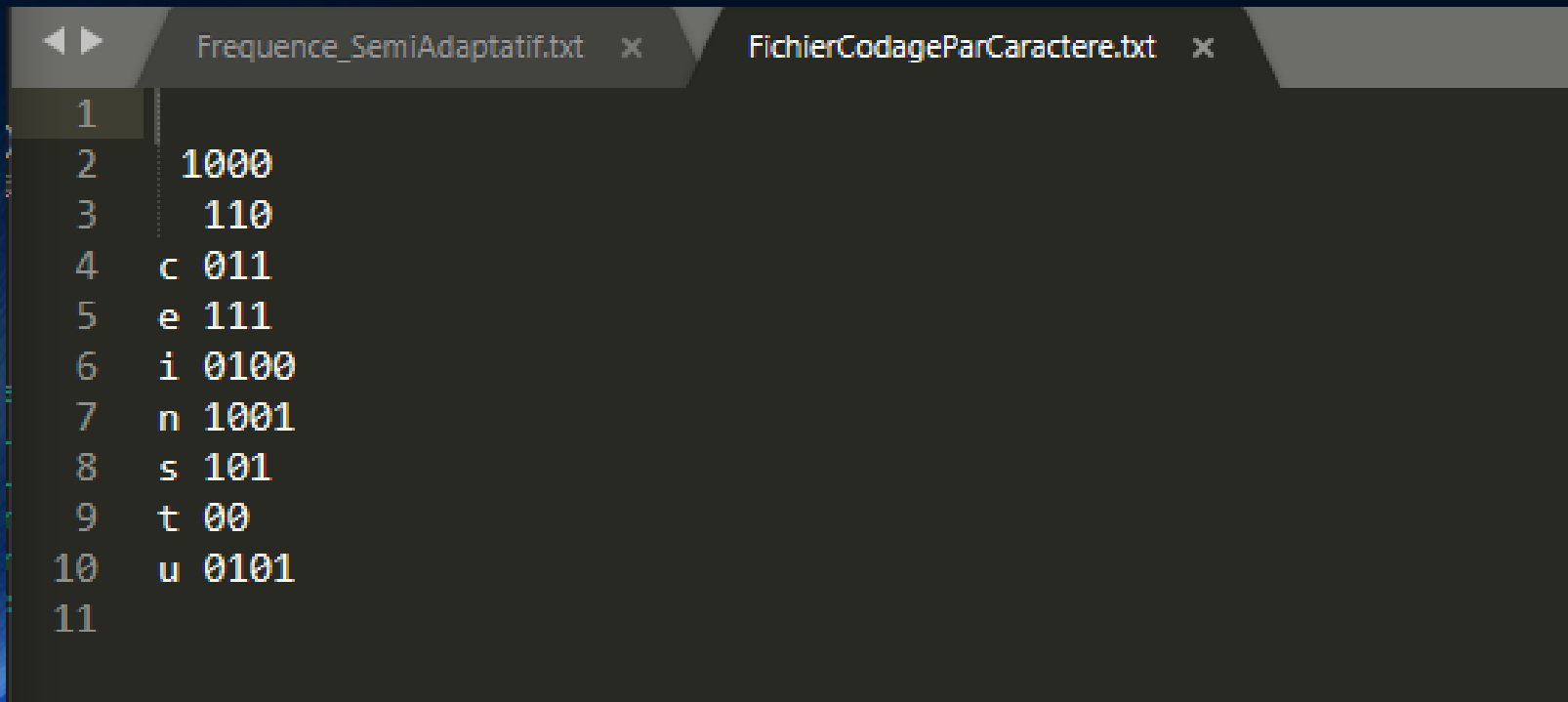
2) Création d'un arbre binaire de recherche basée sur la fréquence d'apparition des caractères : plus un caractère est présent, plus sa place dans l'arbre est haute (les nœuds représentant les caractères sont obligatoirement des feuilles)



EXEMPLE DE FONCTIONNEMENT

HUFFMAN SEMI-ADAPTATIF

3) 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.

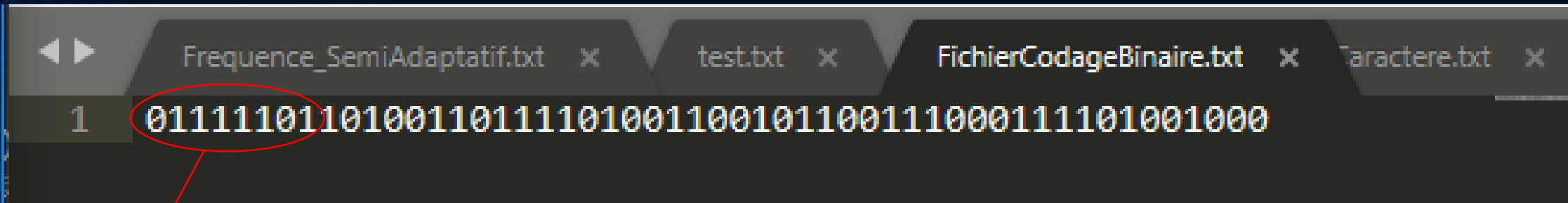


```
1  
2 1000  
3 110  
4 c 011  
5 e 111  
6 i 0100  
7 n 1001  
8 s 101  
9 t 00  
10 u 0101  
11
```

EXEMPLE DE FONCTIONNEMENT

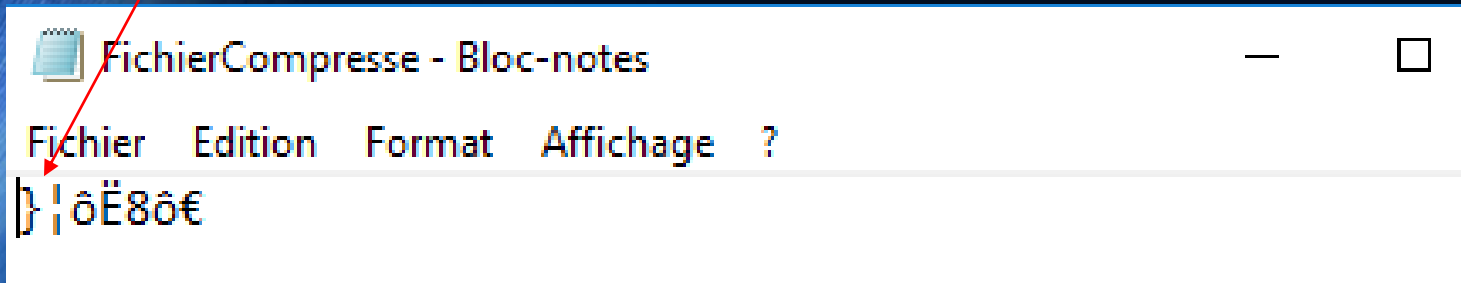
HUFFMAN SEMI-ADAPTATIF

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



```
1 0111110110100110111101001100101100111000111101001000
```

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



```
FichierComprime - Bloc-notes
```

```
Fichier  Edition  Format  Affichage  ?
```

```
}|øË8ø€
```

PRESENTATION DU LIVRABLE

- Exécution en console : menu à 4 options

```
Main (2) [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (29 avr. 2019 à 20:11:14)
```

```
Veillez selectionner une option :
```

- ```
1 - Coder un fichier avec Huffman statique
2 - Coder un fichier avec Huffman semi-adaptatif
3 - Decoder un fichier .txt
4 - Quitter le menu
```

```
Votre choix :
```

# PRESENTATION DU LIVRABLE

- 1<sup>er</sup> choix : codage avec Huffman statique

```
Main (2) [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (29 avr. 2019 à 20:12:53)
Veuillez selectionner une option :
1 - Coder un fichier avec Huffman statique
2 - Coder un fichier avec Huffman semi-adaptatif
3 - Decoder un fichier .txt
4 - Quitter le menu
Votre choix : 1
Vous avez choisit de coder un fichier .txt en Huffman statique.
Nom du fichier .txt dans le répertoire du projet : extrait_alice.txt
Nom du fichier .txt contenant les fréquences pour le codage : frequence_statique.txt
...conception du code binaire associé au texte en cours...
Conception et compression terminées.
```

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
- FichierComprime.txt : fichier résultat de taille plus faible



# PRESENTATION DU LIVRABLE

- 2<sup>ème</sup> choix : codage avec Huffman semi-adaptatif

```
Main (2) [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (29 avr. 2019 à 20:12:53)
```

```
Veillez selectionner une option :
```

- ```
1 - Coder un fichier avec Huffman statique  
2 - Coder un fichier avec Huffman semi-adaptatif  
3 - Decoder un fichier .txt  
4 - Quitter le menu
```

```
Votre choix : 2
```

```
Vous avez choisit de coder un fichier .txt en Huffman semi-adaptatif.
```

```
Nom du fichier .txt dans le répertoire du projet : extrait_alice.txt
```

```
...conception du code binaire associé au texte en cours...
```

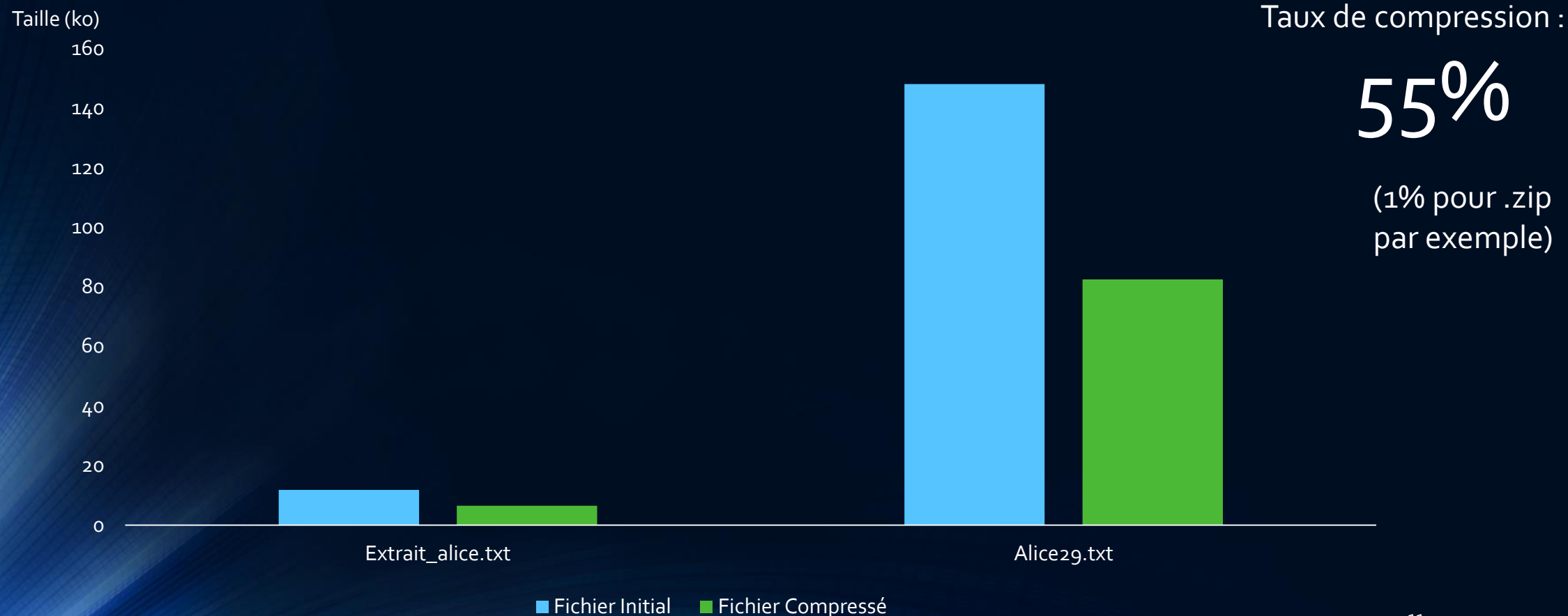
```
Conception et compression terminées.
```

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 tout les caractères présents dans le texte ainsi que le nombre d'apparition.

```
FichierCompreste - Bloc-notes
Fichier  Edition  Format  Affichage ?
R?      ?M?DePyB?/?Üº[]]æÐ?¹Èa▲¿.}?5!)?@@      Ðé°-è¢[]iIJ      [v]??w|òùt!lgBDD¢??ÓQ[]~'òL»/Î]ÎËa []|&[]¥(      ??ÑÂ5Üº[]3?      ÈÚ~??B~?&i*0
I!5äçB?E£ÜG-ÇÂ<Îµ?¥?ny?Ä!ÿ▲|?â?¶Í?[]-NÆ°P[]:[]C[]N      o?1;[]é@°RBë&Õ|s?±[]>ìÂ])/R?.Ý$`´â°???F[]b"}Ü@[]???:R3ü[]?U(0      1[]NR<ÌòìÜC@^Öáôì8«¿dÜ~?Ä?;??%±;
%?ætØ?/?eiâ_eÊPB<v»÷(?°[]:·øð%þ%(g[]7çø7[]?iR      yHÆëKRì.[]Èa=[]??sâÖÖ[]"??Ïþ#wîR5Ü±1FËpPXë7ë-9Y¹³K?G²$?2-Ü[]?%      Î=[]j¿Sâ~°]N(ÿ{Ìò?íRg*Ó[]???ì
?iÜà4?I?L65&O,Ý?>???%[]??øNyú[]N?±Äv[]Sè[]µ$`~³3ëO_R?æZp¹MBS[]«äÊ°*[]]ÆFþ-n´°Dø[]?[]-?Uóv%[]S±ÄO£íc?2Ýü-??Iµk?=O[]w~Ê9n>%,|ZqVÿ:GüÊ?D)æ-ßÈÜÜ[Ï?
[]~Q*Ú;[]¢¢ÉÖïëÿ?[]1RDP[]iµ[]?â[]Èh2-;*$?´Bèò?æ#æÍBé?{éEIHöQËqð?3'WçËbÓ[]?Z=?ý???:¥ZoðèBçÝ2°C[]ç?¢6¶fÖ·?¥%J[]Òìq°â°e±?±æ,BÊ?w¿Öd?´â[»]?[]fÂ&?Òr
???GÜ;¿(?i?üiÜ[]u¥#þjNÁ[]ÇÆLn`¥ýL6øðZqhPV?Z=?N*]?j?N1°ÖÖ-Ñ^      1[]¶ÿYhPV?§ÿÝÜ.¿6þø`{§Üý(?B[]i~U[]P5t?Öv[]b?IÄO»é?6[]è?·ÿ/´[[]i7[]mâ$ð«mp²ðÖc[]??â      ø[]?
ìs?±[]¥?R#x?HBSUI,Ð´â!ÄöPË°xOG.öUøä'??,<?¢[]|Cé?ù?Äþð%T[]=)Rg{??{7tù4ù?G-ÇÔ?ö7[]][]x};n+ÄTIñ_b?g[]¿ÌsÆ?§ü/[]@[]ÏÜìYáö?,´Q¥?RìRg[]¿ÈñÝ?^~}?Ó?³cD4
y«+c$E$[]%¹Ä>Í'°?âi~?Aÿ1[]¢[]?ø?B¹IZîZ?]üìð?±ÚiP?a¿Pð5_      î¿XQpb5CNe$*{ÇG{oè«r?&h~µ(      ?ÓgQ[]w¿P?s°%-t¢?nZ»I?z?az?      !yu*»[]?a=[]¥@      []?JG?°?
?IÄ>Íd[]ÿìð?ø×J[]â"Ó?ÜÜi8«¿c|·?°=þµÄ"eüøµ?>=a[]j?KÎi?ø?Ö[]-P?10%°£üÊ»Wø?Ä«Mp%[]wìo?§?±öy`KÄ80¿>Á.▲ÿyüdiü[]?ëé?ú-8þèKò?Óu?qi¿Ä[]ÖE8´âN·[]'|Î+k
%ýÊ#Kn[ýd(ðk×iJ[]PÖ5æ8`mpøäíe$[]YieM?è?f?s33`£?äèGüÖ?¿¢[]i[]nRé?nð%|hzB[]F?]?»ð7é(g[]y«2qè%N?]?[]i[])R?æg[]G?]?|þ-?ø[]Ì?óðBËG°3C?]?iâÉFó]Hÿ???Çú?²ðÜ[]N
?[]z4?i÷2ÖµÄM?èNÄ[]5?¿óféò?æoäÿ-;Y?iÝ&³Iò?³IÄO£IÄ°?¿4BÍüÖÊÊ?ü[]ÄÄÄÆ-[?9Sè`{4?#ÎÖ-ÊÖ[]<¥B7úbó?sòP[]iBËGµFúâ8?þVÆG?[]?#[]¹TñSìðq§ÜËa;[]é@%=>?[??
[]°[[]oðÊ·h7øhP?nî\°?µI?Ë7f?s>Ö?ýkÖ      ;[]üü´nÉ·kpìóóìjBó2ix-é)öZ7?¥?rÜ}è[]3?7Çiâ7~ð9E1B?[]?Íu$1·?ø*Ý      ¹Ü¥·X´âQøKÎÄa[]°óú(øµ«v?ÊBv[]      j¥Ó~?¿MèÜ[]RQ
?ýUøµó[]rÜ|UB±%s×ø?K?      °???7¥kÊðSû??nî\M«?Ï?%Ä?æ.Î4?ä!Kþø¶4?Q[]ÿeU[]-ÝË?¿Ýiy[]]L%Æ7øµ?%[]!ý³µ«v?%&%?      7ë#ø[]?Y0Ö|GP=aGù?¿r?[]\hZoðè0;äMáì?´´
Q[]nNV?µ#èÜ[]?y?þÖ|´â???ìÜ%3¿>èòì8G?;|/|9&ýcâ¿[]iÜV[]iN=3b*J]??²O«/Î[]|'kÉC???æTPJc~Q[]x0(g[]°ò?´´[]?7?¿ù?y«!ý²uBçB      §üÈMè£?äâ!ðS>Ö<µÖ[]nN%²Ü-p±
Ü|ZoD%çjðBµÈâ?æ1%?¢*iøB¥T=VVTçænÇ?~òù%Ä±[[]nÐË$[â£´éoð?ÂÝÖ.w)mÖSpcA[]M-Zq[]~`[]Îâ-°PðFB?¿øðAXjÊ?JY§ø{âá?|²Ó?ýM.ü|4      oD      øYÿ~`T[]¶"|q[]x³úì~
t?2Ï?T?#è0¿é{R?u7¢[]óµ{o?¿eì6#o%3ícÿUv·hPXâxç²|ð[]Z7?¥ò?æñÍ=?6[]$B~½-Ä[´[]ÜÄx×ø3cµ°?æfú·z      1rsÌòûN36·{o?ùRn4"o[]5äÊQ[]i=Dx}-ÜøèU¿?¿fÜwæ;C@&Ö
```

PRESENTATION DU LIVRABLE

Etude de la taille des fichiers à coder



PRESENTATION DU LIVRABLE

- 3^{ème} choix : décodage fichier encodé par Huffman avec valeur binaire de chaque caractère

```
Veuillez selectionner une option :  
1 - Coder un fichier avec Huffman statique  
2 - Coder un fichier avec Huffman semi-adaptatif  
3 - Decoder un fichier .txt  
4 - Quitter le menu  
Votre choix : 3  
Vous avez choisit de décoder un fichier .txt en Huffman.  
Nom du fichier .txt dans le répertoire du projet : FichierCompresse.txt  
...décompression et décodage en cours...  
Décodage et décompression terminés.
```

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

PRESENTATION DU LIVRABLE

- 3^{ème} choix : décodage fichier encodé par Huffman avec valeur binaire de chaque caractère

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

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 bahCoa 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, foi e hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getti up and picki the daisies, when sudde,gtfi ite Rabbit with pink eyes ran close by her.



There was nothi 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

52 différences
2203 mots

98%

ASPECTS TECHNIQUES DU DEVELOPPEMENT

- 1 classe Main sur Java et 3 classes auxiliaires :

 Huffman	25/04/2019 16:26	Fichier JAVA	5 Ko
 Main	29/04/2019 20:18	Fichier JAVA	8 Ko
 Node	02/04/2019 12:20	Fichier JAVA	2 Ko
 ReadFileToCompress	26/04/2019 14:29	Fichier JAVA	3 Ko

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

ASPECTS TECHNIQUES DU DEVELOPPEMENT

- Classe Huffman.java :

Huffman
PriorityQueue<Node> nodes TreeMap<Character, String> codes String textToEncode String encodedText Hashtable<Character, Integer> dico
----- encodeText() : String buildTree(PriorityQueue<Node>): void frequency(PriorityQueue<Node>): String createNodes(" , String file): void encodingText(Node , String): void

```

Huffman.java x Main.java Node.java ReadFileToCompress.java
1 import java.io.BufferedReader;
10
11 public class Huffman {
12
13     private PriorityQueue<Node> nodes = new PriorityQueue<>((o1, o2) -> (o1.getValue() < o2.getValue()) ? -1 : 1);
14     private TreeMap<Character, String> codes = new TreeMap<>();
15     private String textToEncode = "";
16     private String encodedText = "";
17     private Hashtable<Character, Integer> dico = new Hashtable<Character, Integer>();
18
19
20     public Huffman(String textToEncode){
21         this.textToEncode = textToEncode;
22     }
23
24     public String getTextToEncode() {
25         return this.textToEncode;
26     }
27
28     public PriorityQueue<Node> getNodes(){
29         return this.nodes;
30     }
  
```

PriorityQueue : liste classé 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.

ASPECTS TECHNIQUES DU DEVELOPPEMENT

- Classe Huffman.java :

```
Huffman

PriorityQueue<Node> nodes
TreeMap<Character, String> codes
String textToEncode
String encodedText
HashMap<Character, Integer> dico

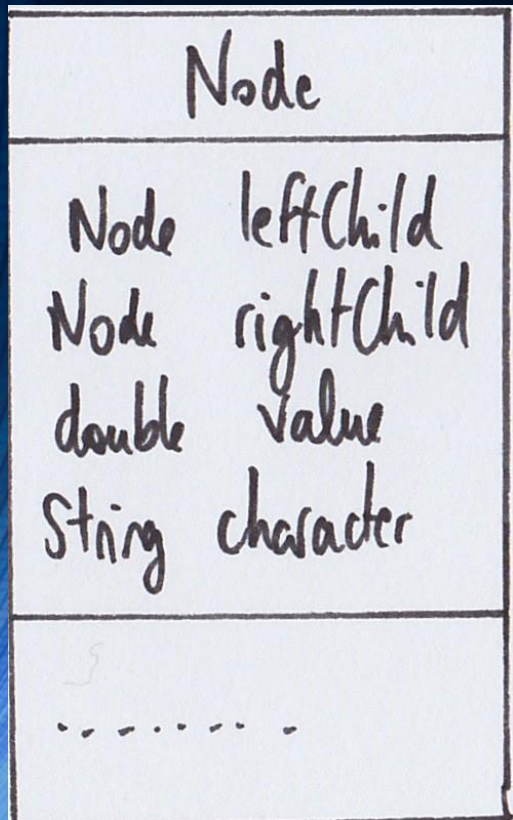
-----

encodeText() : String
buildTree(PriorityQueue<Node>): void
frequency(PriorityQueue<Node>): String
createNodes( " , String file ): void
encodingText(Node, String): void
```

- frequency(...) : méthode qui retourne un String avec la fréquence d'apparition de chaque caractère (ce que l'on trouve dans le fichier frequency_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 mets dans la liste nodes.
- buildTree(...) : une fois tout les nœuds créés, 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 au final un nœud père avec des fils successifs (principe de l'arbre binaire).
- encodingText(...) : on parcourt tout les nœuds à partir du nœud racine, en retenant 0 pour les fils gauche et 1 pour les fils droit, 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.

ASPECTS TECHNIQUES DU DEVELOPPEMENT

- Classe Node.java :



```
Huffman.java Main.java Node.java x ReadFileToCompress.java
2 public class Node {
3
4     private Node leftChild;
5     private Node rightChild;
6     private double value;
7     private String character;
8
9     public Node(double value, String character) {
10
11         this.value = value;
12         this.character = character;
13         this.leftChild = null;
14         this.rightChild = null;
15     }
16
17     public Node(Node left, Node right) {
18         this.value = left.value + right.value;
19         character = left.character + right.character;
20         if (left.value < right.value) {
21             this.rightChild = right;
22             this.leftChild = left;
23         } else {
24             this.rightChild = left;
25             this.leftChild = right;
26         }
27     }
28 }
```

ASPECTS TECHNIQUES DU DEVELOPPEMENT

- Classe ReadFileToCompress.java :

ReadFileToCompress
String fileName
readFile(): String readCharacterFile(): String getDicoCodage(): Hashtable<Character, String>

```
10 public class ReadFileToCompress {  
11  
12     private String fileName;  
13  
14     public ReadFileToCompress(String fileName) {  
15         this.fileName = fileName;  
16     }  
17     public String readFile() throws IOException {  
18         String text = "";  
19         BufferedReader br = new BufferedReader(new FileReader(fileName));  
20         String line;  
21         while ((line = br.readLine()) != null) {  
22             text = text + line + "\n";  
23         }  
24         br.close();  
25         return text;  
26     }  
}
```

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