DORMOY Alexis, HABCHI Sonia, MAQSOOD Shahzeb

HABCHI Sonia

MAQSOOD Shahzeb

Compression et decomPression selon l’algorithme de HUFFMAN

STRUCTURE DE DONNEES

Sommaire :

[Résumé 1](#_Toc33801650)

[Introduction 1](#_Toc33801651)

[Problème traitée 1](#_Toc33801652)

[Plan du document 2](#_Toc33801653)

[Rappel succinct 2](#_Toc33801654)

[Architecture de l’application 2](#_Toc33801655)

[Diagramme des classes 2](#_Toc33801656)

[Principaux algorithmes 2](#_Toc33801657)

[Mise au point 2](#_Toc33801658)

[Explication 2](#_Toc33801659)

[Exemple d’utilisation 2](#_Toc33801660)

[Difficultés rencontrées et solutions choisie 2](#_Toc33801661)

[Conclusion 2](#_Toc33801662)

[Etat d’avancement 2](#_Toc33801663)

[Perspectives d’évolution 2](#_Toc33801664)

# Résumé

Ce rapport contient une introduction avec un rappel du problème énoncé, le plan du rapport, un rappel des conditions de développement, l’architecture de l’application ainsi que les principaux algorithmes détaillés et expliqués. Une mise au point du projet ainsi que les difficultés rencontrés et les solutions envisagés (et surtout retenu) pour gérer ces problèmes.   
Il contient également une conclusion sur l’avancement et l’état actuel du projet ainsi que ses perspectives d’évolution.

# Introduction

## Problème traitée

Un fichier texte est composée de caractères, ces caractères sont codés sur un ensemble de bits. Le plus souvent on utilise la norme ASCII qui permet de codé 128 caractères. Chaque caractère est codé sur 1 octet (8bits) de 0000 0000(0) à 1111 1111(127) et donc chaque caractère prend la même place en mémoire a savoir 1 octet.  
Nous cherchons donc à savoir s’il n’y a pas un moyen d’écrire du texte en prenant moins de place en mémoire et donc de compresser l’écriture des caractères en diminuant le nombre de bits nécessaire et donc réduire la taille en octet d’un fichier texte en le compressant. Il faut toutefois être capable de le décompresser par la suite, la compression doit donc être sans perte.

## Plan du document

Notre document prendra donc la forme suivante : un rappel du sujet qui correspond à la solution choisi pour la compression d’un fichier texte ainsi qu’une brève explication de son fonctionnement, de ses avantages et inconvénients. Puis de l’architecture de notre application pour voir l’articulation de l’application et les liens entre les fichiers ainsi que leur utilités .  
Une explication détaillé des principaux algorithmes pour comprendre leurs principes

Une explication du fonctionnement de l’application et des exemples d’utilisation.  
Les différents problèmes rencontrés lors du développement , les différentes solutions envisagées et celle retenue.

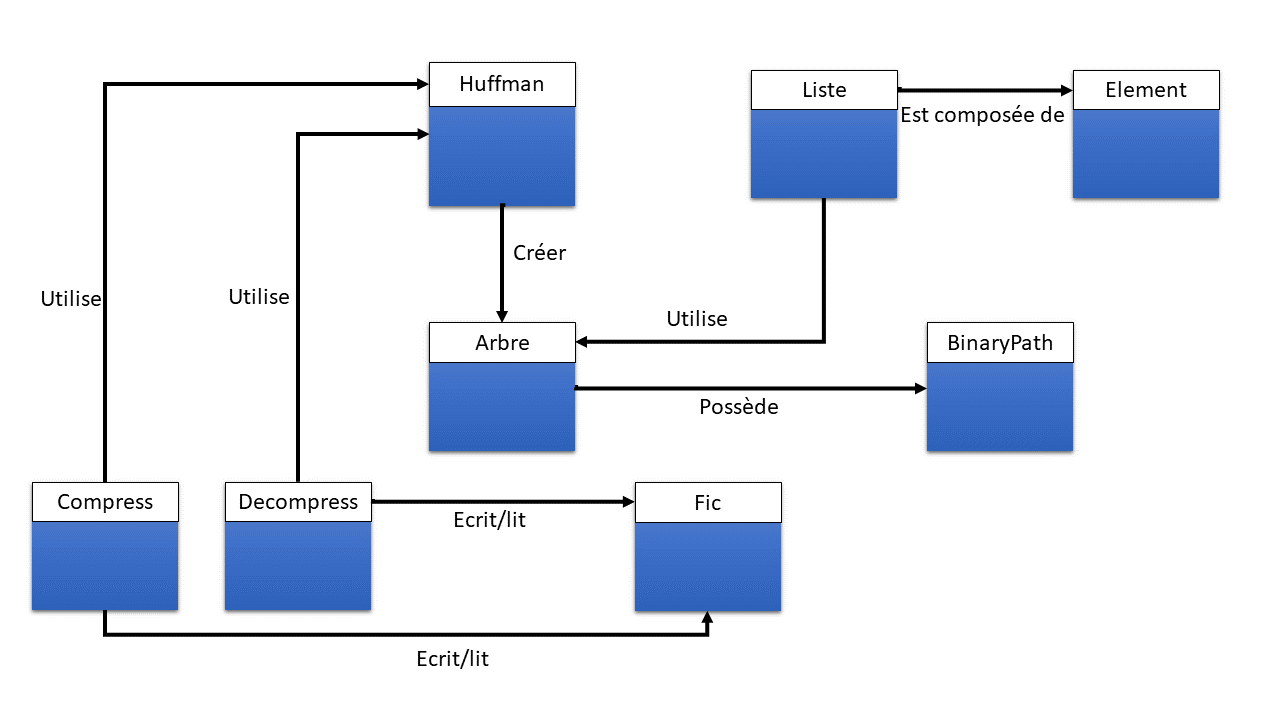
Une conclusion sur l’avancement du projet et ses évolutions possibles.

# Rappel succinct

Nous devons réaliser une application de compression/décompression de fichier texte en suivant l’algorithme de HUFFMAN.  
Cet algorithme est basé sur la création de l’arbre et de la table de HUFFMAN ainsi que sur la lecture préalable du document avant sa compression. En effet, nous devons connaître les caractères présent dans le document ainsi que leur fréquence pour construire notre algorithme. Plus un caractère est fréquent, plus on l’associera a un code binaire court et à l’inverse un caractère peu présent sera associé à un code binaire plus long.

# Architecture de l’application

## Diagramme des classes



# Principaux algorithmes

# Mise au point

## Explication

## Exemple d’utilisation

# Difficultés rencontrées et solutions choisie

# Conclusion

## Etat d’avancement

## Perspectives d’évolution