

RAPPORT TECHNIQUE SUR LE PROJET OXPRESS

I. Description

Oxpress est une application CLI (Command Line Interface) développée en Rust, permettant la compression et la décompression sans perte de fichiers texte de volume plus ou moins grande ($V(f) \geq 100\text{Mo}$).

L'objectif du projet est d'implémenter et d'étudier des algorithmes fondamentaux de compression de données, tout en garantissant :

- Une reconstruction bit-à-bit identique du fichier original
- Une gestion correcte des fichiers volumineux
- Une interface terminal élégante avec affichage des statistiques

Le programme fonctionne exclusivement sur des fichiers texte, et assure une décompression parfaitement fidèle grâce à la gestion explicite de la taille originale des données.

II. Algorithmes utilisés

Le système repose sur la combinaison de deux algorithmes classiques :

1. LZ77 (Lempel-Ziv 1977)

LZ77 est un algorithme de compression basé sur la détection de répétitions dans une fenêtre glissante.

Il remplace les séquences répétées par des triplets : (offset, longueur, caractère_suivant)

Son rôle dans oxpress est :

- Supprimer les redondances locales
- Réduire les répétitions fréquentes dans le texte

2. Codage de Huffman

Le codage de Huffman est un algorithme de compression entropique basé sur :

- La fréquence d'apparition des symboles
- La construction d'un arbre binaire optimal

Son rôle dans le programme est :

- Encoder efficacement les données produites par LZ77
- Réduire davantage la taille en utilisant des codes binaires variables

3. Pourquoi la combinaison LZ77 + Huffman ?

La combinaison permet :

- LZ77 : suppression des répétitions
- Huffman : compression statistique des symboles restants

Cette approche est similaire aux principes utilisés dans des formats réels comme **ZIP ou DEFLATE**.

III. Architecture du système

Le pipeline de compression suit les étapes suivantes :

1. Compression

- Fichier texte original
- Compression LZ77
- Sérialisation des données
- Sérialisation des données
- Compression Huffman
- Écriture du fichier compressé avec en-tête

Le fichier compressé contient :

- La taille exacte du fichier original
- Les données encodées
- La structure de l'arbre de Huffman

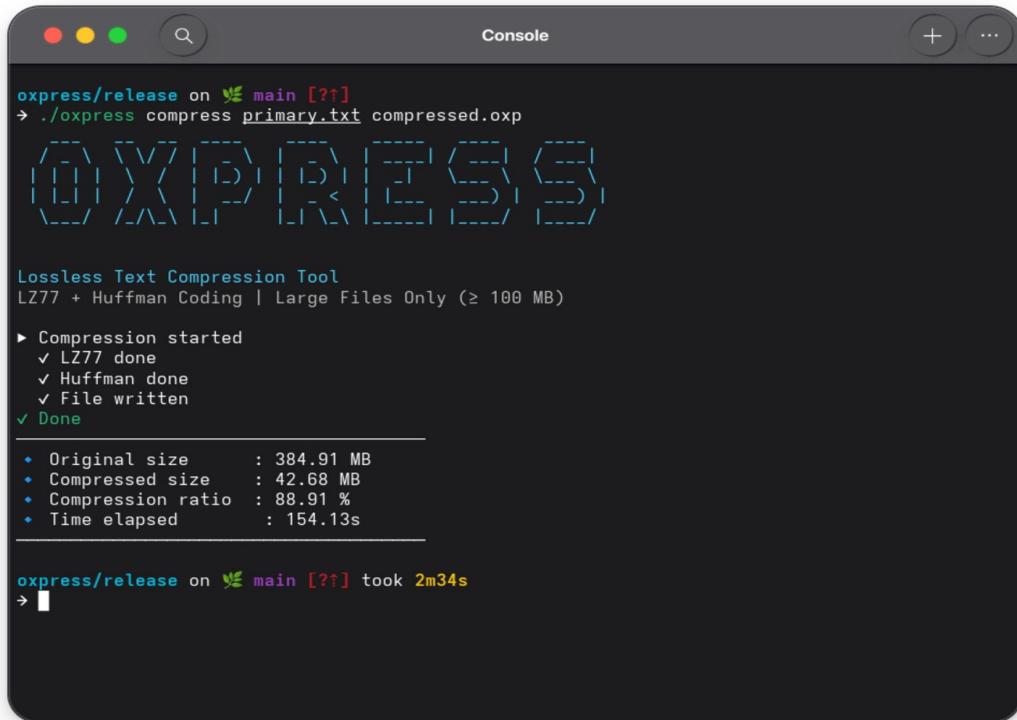
2. Décompression

- Fichier compressé
- Lecture de la taille originale
- Décompression Huffman
- Reconstruction des données LZ77
- Décompression LZ77
- Troncature à la taille exacte originale : permet d'éliminer les bits de padding ajoutés lors de l'alignement binaire
- Fichier restauré

IV. Résultats expérimentaux

Des tests ont été effectués sur plusieurs fichiers texte.

Pour une exemple, on a choisi un fichier de taille de 382.



The screenshot shows a macOS terminal window titled "Console". The command entered is:

```
oxpress/release on ➜ main [?↑]
→ ./oxpress compress primary.txt compressed.oxp
```

Below the command, there is a large block of binary data represented by various symbols like '|', '(', ')', '{', '}', '[', ']', and '/'. This is the compressed file content.

Following the compressed data, the tool's status message is displayed:

```
Lossless Text Compression Tool
LZ77 + Huffman Coding | Large Files Only (≥ 100 MB)

▶ Compression started
  ✓ LZ77 done
  ✓ Huffman done
  ✓ File written
✓ Done
```

A summary of the compression results is provided:

```
• Original size      : 384.91 MB
• Compressed size   : 42.68 MB
• Compression ratio : 88.91 %
• Time elapsed       : 154.13s
```

At the bottom, the terminal shows the completion message:

```
oxpress/release on ➜ main [?↑] took 2m34s
→ █
```

```
express/release on ✌ main [?] 
→ ./oxpress decompress compressed.oxp decompressed.txt
[Progress bar: 0% to 100%]
Lossless Text Compression Tool
LZ77 + Huffman Coding | Large Files Only (> 100 MB)
▶ Decompression started
✓ Archive loaded
:: Decoding Huffman...
```

Après décompression :

- Vérification via SHA-256
- Les empreintes sont identiques
- Reconstruction parfaite du fichier

```
express/release on ✌ main [?] 
→ sha256sum primary.txt decompressed.txt
04a87b6255af600e79032b474751920a563bd888dce997409d9cbe143e61414b  primary.txt
04a87b6255af600e79032b474751920a563bd888dce997409d9cbe143e61414b  decompressed.txt

express/release on ✌ main [?] took 3s
→ █
```

Formule utilisée pour le taux de compression :

$$\text{Taux (\%)} = (1 - (\text{taille_compressée} / \text{taille_initiale})) \times 100$$

V. Analyse et Discussion

1. Performance

La combinaison LZ77 + Huffman offre :

- Un bon taux de compression sur les fichiers texte
- Une efficacité accrue lorsque le texte contient des répétitions fréquentes

Les performances dépendent fortement :

- De la taille de la fenêtre LZ77
- De la distribution statistique des caractères

2. Limites

- Plus la taille du fichier est grande, plus le programme prend du temps pour le compresser
- Puisqu'on a utilisé LZ77 comme première algorithme dans notre pipeline, le programme ne compressera pas correctement et efficacement s'il n'y a pas de répétitions fréquentes dans le fichier à compresser

3. Problèmes rencontrés

Durant les phases de test :

- Gestion des bits de padding dans Huffman
- Correction via stockage explicite de la taille originale

Ce correctif garantit une décompression parfaitement fidèle.

VI. Conclusion

Oxpress démontre l'efficacité de la combinaison LZ77 + Huffman pour la compression sans perte de fichiers texte.

Le projet met en évidence :

- Les principes fondamentaux de la compression
- L'importance de la gestion des flux binaires
- Les subtilités liées au padding et à la reconstruction exacte

Ce travail constitue une base solide pour des nouvelles perspectives innovante comme l'ajout d'optimisations avancées ou encore l'implémentation d'autres algorithmes (BWT, Arithmetic Coding) sans oublier l'amélioration des performances.