INF8770 – Technologies multimédias

Hiver 2023

Travail pratique 1

Groupe 2

2073519 – Charles-Antoine Vézina
2017113 - Yun Ji Liao

5 Février 2023

Question 1 :

Pertinence sur les images :

I**mages 1 :** Il n'est pas pertinent d'utiliser le codage de Huffman parce que tous les pixels sont de couleur unique. Donc, les couleurs ont la même fréquence d'apparition. Le codage de Huffman dépend de la fréquence d'apparition pour définir les bits et les sauvegarder dans une table.

Images 2 :

Vu que l'image est une seule couleur, il est beaucoup plus pertinent d'utiliser le codage par plage au lieu de codage de Huffman. En effet, Huffman dépend du nombre d'apparitions de chaque bit utilisé. Pour cette image, il est inutile de le connaître vu qu'ils sont de la même couleur. Donc, Huffman n'est pas efficace et il n'y aura aucune différence après la compression.

Images 3 : Le codage de Huffman sera pertinent vu que l'image a des couleurs différentes qui se répètent. Intuitivement, on peut voir qu'il y a du blanc et de différents types de rouges qui se répètent. Le temps d'exécution est court parce que c'est une image simple. Donc, il est efficace de coder en se basant sur le nombre d'occurrences de chaque couleur.

Images 4 : L'image pourrait être codée en Huffman, mais peu efficace vu qu'il contient beaucoup de blanc et de noir. Par contre, vu que beaucoup de blancs et de noir ne sont pas tous pareils, il faut utiliser de différents bits pour les représenter, mais on voit intuitivement qu'il y a quand même de couleur répété dans l'image. Avec une image complexe, le temps d'exécution sera plus long. Donc, utiliser le codage de Huffman est peu pertinent.

Images 5 : Il est peu pertinent d'utiliser Huffman vu qu'il y a 2 seules couleurs, c'est-à-dire que le codage est basé sur 2 bits. Donc, il n'est pas efficace d'utiliser la compression de Huffman, comme au texte 1.

Pertinence sur les textes :

Texte 1 : Puisqu'il y a seulement deux caractères différents dans le texte, le codage de Huffman est inefficace dans cet exemple, car les deux caractères seront chacun de 1 bit.

Texte 2 : Le codage de Huffman est pertinent dans cette situation puisque le texte continent 3 caractères et que le A est répété fréquemment, la compression avec Huffman permettra de réduire la longueur du message.

Texte 3 : Le codage de Huffman est optimal pour ce texte puisqu'il y a plusieurs symboles, et la lettre O est très répétée alors que les autres lettres ne sont pas souvent représentées. Le codage optimisera la longueur de la lettre O afin d'obtenir un message beaucoup plus court.

Texte 4 : Ici, le codage de Huffman est utile, mais peu pertinent puisque le message est long et qu'il y a des répétitions de plusieurs caractères (' ', 'e', 'i), il est possible de réduire l'espace utilisé par ces bits afin d'avoir un message optimisé.

Texte 5 : Ce texte est similaire au texte 4, donc les mêmes notions vont s'appliquer. Il y a plusieurs répétitions des mêmes caractères et il sera possible avec le codage de Huffman d'optimiser la longueur de ces caractères. Étant donné la longueur du texte, il va y avoir une grande différence entre la longueur du texte initial et celle après compression et le temps d'exécution sera plus long.

Question 2 :

Résultats sur les images :

I**mages 1 :**

Après la compression, le taux de compression est de 0.202 et le temps de compression nécessaire était de 2 710 ms. Avec ces résultats, on remarque que la compression a fonctionné, mais elle a été peu efficace, comme nous avions prédit à la question 1.

```
Longueur = 1409075
Longueur originale = 1766808.0

Temps moyen d'exécution: 2710 ms

Taux de compression : 0.202
```

Images 2 :

Comme nous avions remarqué à la question 1, le codage de Huffman a été inutile sur cette image puisqu'elle ne possédait qu'une seule couleur. On ne tire donc pas avantage de la compression de Huffman et on obtient un taux de compression de 0 et un temps d'exécution de 1 505 ms.

```
Temps moyen d'exécution: 1505 ms

Longueur = 393216
Longueur originale = 393216

Taux de compression : 0.0
```

Images 3 :

Le codage de Huffman a été efficace pour l'image 3, avec un taux de compression de 0.315, mais un temps d'exécution de 14 824 ms, beaucoup plus lent que les premières images. Ces résultats correspondent à ce que nous avions prédit à la question 1, et l'image a pu tirer avantage de cette méthode de compression avec une majorité de blancs et de rouges afin de réduire leur longueur.

```
Temps moyen d'exécution: 14824 ms

Longueur = 4870587
Longueur originale = 7114632.0

Taux de compression : 0.315
```

Images 4 :

Nous avions prédit que l'image 4 ne serait pas très compressée étant donné les différentes couleurs présentes dans l'image. Avec un taux de compression de 0.219 et un temps d'exécution de 15 525 ms, le codage de Huffman a été peu efficace, mais plus que ce à quoi nous nous attendions.

```
Temps moyen d'exécution: 15525 ms

Longueur = 6462731
Longueur originale = 8272876.0

Taux de compression : 0.219
```

Images 5 :

Avec ce que nous avions dit à la question 1, nous nous attendions à ce que le codage de Huffman soit inefficace pour l'image 5 puisqu'elle contient seulement 2 couleurs, il serait donc impossible d'avoir un code optimal. Par contre, avec un résultat de taux de compression de 0.592 et un temps d'exécution de 4318, nous avons regardé les valeurs de couleurs obtenues en décodant l'image et nous avons remarqué qu'il y a au moins 3 couleurs différentes dans l'image. Ceci rend le codage de Huffman possible et efficace pour cette image, car elle possède la couleur noire en grande majorité.

```
Temps moyen d'exécution: 4318 ms

Longueur = 1146392
Longueur originale = 2809216.0

Taux de compression : 0.592
```

Résultats sur les textes :

Texte 1 : Le texte 1 obtient un temps moyen d'exécution de 126 ms et un taux de compression de 0,0.

```
Longueur = 84
Longueur originale = 84.0

Temps moyen d'exécution: 126 ms

Taux de compression : 0.0
```

Le résultat est conforme à la valeur attendue. Le taux de compression sera évidemment de 0,0 puisqu'il est codé sur seulement 2 bits.

Texte 2 : Le texte 2 obtient un temps moyen d'exécution de 129 ms et un taux de compression de 0,234.

```
Longueur = 118
Longueur originale = 154.0

Temps moyen d'exécution: 129 ms

Taux de compression : 0.234
```

Nous croyions que Huffman serait pertinent, mais on aperçoit que le taux de compression n'est pas proche de 1. En conclusion, il est possible d'utiliser Huffman, mais vu qu'on a seulement 3 bits pour mesurer la fréquence d'apparition, il est peu pertinent.

Texte 3 : Le texte 3 obtient un temps moyen d'exécution de 152 ms et un taux de compression de 0,358.

```
Longueur = 131
Longueur originale = 204.0

Temps moyen d'exécution: 152 ms

Taux de compression : 0.358
```

Le résultat est conforme à nos remarques. Le codage optimise la longueur de la lettre utilisée fréquemment, ce qui résulte d'un message plus court.

Texte 4 : Le texte 4 obtient un temps moyen d'exécution de 212 ms et un taux de compression de 0,187.

```
Temps moyen d'exécution: 199 ms

Longueur = 739
Longueur originale = 910.0

Taux de compression : 0.188
```

Le résultat est conforme à nos remarques puisque Huffman pourrait être utilisé pour la compression, mais le taux de compression est bas.

Texte 5 : Le texte 5 obtient un temps moyen d'exécution de 315 ms et un taux de compression de 0,3.

```
Temps moyen d'exécution: 275 ms

Longueur = 5601
Longueur originale = 7998.0
Taux de compression : 0.3
```

Le résultat est attendu.

Question 3 :

Nous avons proposé le codage LZW. En effet, trouver la meilleure méthode de codage pour optimiser la compression pour tous les textes et images est difficile. Chaque texte et image est unique et peut être optimisé par des méthodes différentes. Nous posons l'hypothèse que le codage LZW pourrait être meilleur que la compression de Huffman en général parce qu'il existe souvent des répétitions dans les messages. Donc, on peut se servir d'un dictionnaire pour stocker des séquences et améliorer le taux de compression. D'ailleurs, nous avons pensé à utiliser le codage par plage à cause de certaines de ces répétitions, mais il serait coûteux pour les textes et images de contenant pas de répétition du même symbole.

Texte 1 :

```
Temps moyen d'execution: 51 ms
Longueur = 74
Longueur originale = 84.0
Taux de compression : 0.119
```

Texte 2 :

```
Temps moyen d'execution: 57 ms
```

```
Longueur = 113
Longueur originale = 154.0

Taux de compression : 0.266
```

Texte 3 :

```
Temps moyen d'execution: 95 ms

Longueur = 182
Longueur originale = 204.0

Taux de compression : 0.108
```

Texte 4 :

```
Temps moyen d'execution: 71 ms

Longueur = 909
Longueur originale = 910.0

Taux de compression : 0.001
```

Texte 5 :

```
Temps moyen d'exécution: 327 ms

Longueur = 5870
Longueur originale = 7998.0

Taux de compression : 0.266
```

Image 1 :

```
Temps moyen d'exécution: 6180 ms

Longueur = 72431
Longueur originale = 1560600.0 Taux de compression : 0.954
```

Image 2 :

```
Temps moyen d'exécution: 2267 ms
Longueur = 7848
Longueur originale = 393216

Taux de compression : 0.98
```

Image 3 :

```
Temps moyen d'exécution: 25899 ms

Longueur = 134056
Longueur originale = 5232192.0

Taux de compression : 0.974
```

Image 4 :

```
Temps moyen d'exécution: 52130 ms

Longueur = 191650
Longueur originale = 6000000.0

Taux de compression : 0.968
```

Image 5 :

```
Temps moyen d'exécution: 4127 ms

Longueur = 14637
Longueur originale = 2444550.0

Taux de compression : 0.994
```

Question 4:

Table 1: Résultats des messages compressés par la méthode de Huffman et la méthode de LZW

	Taux de compression Huffman	Taux de compression LZW	Temps d'exécution Huffman(ms)	Temps d'exécution LZW(ms)
Image_1	0,202	0,954	2710	6180
Image_2	0,0	0,980	1505	2267
Image_3	0,315	0,974	14824	25899
Image_4	0,219	0,968	15525	52130
Image_5	0,592	0,994	4318	4127
Texte_1	0,0	0,119	126	51
Texte_2	0,234	0,266	129	57
Texte_3	0,358	0,108	152	95
Texte_4	0,188	0,001	199	71
Texte_5	0,30	0,266	275	327

Au cours de l'expérience, nous avons conclu que le taux de compression de la méthode LZW est meilleur que celui de Huffman pour le texte 1 et 2 et significativement pour toutes les images. Il est possible d'expliquer cette différence dans les textes puisque le texte 1 et 2 présente beaucoup de répétition de séquence de message. La différence entre la compression LZW et Huffman est mince pour le texte 2, mais nous pensons que si le texte était plus long, mais restait similaire, on pourrait mieux voir l'efficacité de la compression LZW. On constate aussi que les images codées avec LZW ont un meilleur taux de compression dû aux répétitions des couleurs RGB, qu'on peut apercevoir lorsqu'on regarde la liste de valeurs de RGB lors de la lecture de l'image.

Pour le temps d'exécution, la méthode LZW est plus efficace dans les textes 1, 2, 3 et 4. Trouver les répétitions dans les images prennent plus de temps parce que la taille est plus grande, ce qui fait que créer le dictionnaire prend plus de temps. Inversement pour les textes, vu qu'ils ont moins de taille, il prend moins de temps à créer le dictionnaire. D'ailleurs, le temps d'exécution varie selon le serveur Google, lorsqu'on remarque deux temps d'exécutions similaires, il est difficile de conclure lequel est le meilleur, puisqu'il peut varier d'une dizaine de millisecondes entre deux essais.

La limitation de notre algorithme est le fait qu'il faut avoir plusieurs séquences de symboles répétés dans le message à compresser pour avoir une bonne compression et l'efficacité est meilleure lorsque les textes sont plus longs. Dans certains cas, il est même possible d'obtenir un message compressé plus gros que l'original (par exemple dans la question 1 de l'intra A22) étant donné que la compression LZW possède un code de longueur variable.

Sources :

Les autres fichiers fournis avec la remise de ce travail sont les codes utilisés afin d'obtenir les résultats présentés dans ce rapport. Les codes sont basés sur les exemples de code disponible sur le site Moodle du cours INF8770. Certaines modifications ont été faites afin d'obtenir plus de donner, tels les temps d'exécutions et les taux de compression. Nous avons utilisé d'autres sites internet comme Stack Overflow afin de trouver des solutions afin de convertir nos images en liste/texte.