INF8770 – Technologies multimédias

Hiver 2024

TP1

Groupe 1

2136374 – Huy Viet Nguyen

2143877 – Thomas Thiboutot

Soumis à : Mehdi Miah

5 février 2024

Question 1

Le codage LZW est pertinent lorsque le message contient plusieurs différentes chaînes de symboles qui se répètent souvent. Il est à noter que les messages doivent être suffisamment longs et avoir suffisamment de chaînes qui se répètent pour permettre “d'accumuler” les gains de compressions. L’inverse étant vrai, le codage LZW n’est pas pertinent quand il n’y a peu ou aucune chaîne qui se répètent. Puisque le codage LZW parcourt le message une fois, la complexité du codage LZW est de l’ordre de O(n). Par conséquent, on peut déduire que le temps de compression d’un message est proportionnel à sa taille.

Pour les textes, le codage LZW semble donc pertinent pour les textes : 1, 2, 3, et 5, car ils ont des chaînes de caractères qui se répètent. Il est à noter que le texte 5 est un cas particulier par rapport aux textes 1 à 3. En effet, le texte 5 semble être composé de mots aléatoires, ce qui semble ne pas être favorable à la compression. Cependant, le texte devrait être suffisamment long pour que les syllabes qui se répètent puissent permettre d’avoir un taux de compression non négligeable. Contrairement au texte 5, le codage LZW ne semble pas pertinent pour le texte 4, car il est relativement court et il y a peu de mots qui se répètent.

Pour les images, le codage LZW devrait être pertinent quand on a des chaînes de pixels qui se répètent. Le codage LZW devrait donc être pertinent pour les images : 2, 3, 5, car on a de grandes sections de l’image qui sont d’une seule couleur. Le codage LZW pourrait aussi être pertinent pour l’image 4, car on observe des sections avec des tons noirs pour les montagnes et des sections avec des tons blancs pour la neige. Le codage LZW ne sera pas pertinent pour l’image 1 parce qu’elle représente un dégradé de couleur, il aura donc très peu de pixels de même couleur, voire même aucun.

Question 2

À partir des données recueillies lors de l'exécution du code, les tableaux ci-dessous illustrent les performances du codage LZW pour des textes et des images.

Tableau 1. Performance du codage LZW sur les textes

Texte	Taux de compression	Temps de compression (ms)
1	0,1190	0,1509
2	0,2662	0,1554
3	0,1078	0,2189
4	0,0011	1,023
5	0,2649	9,847

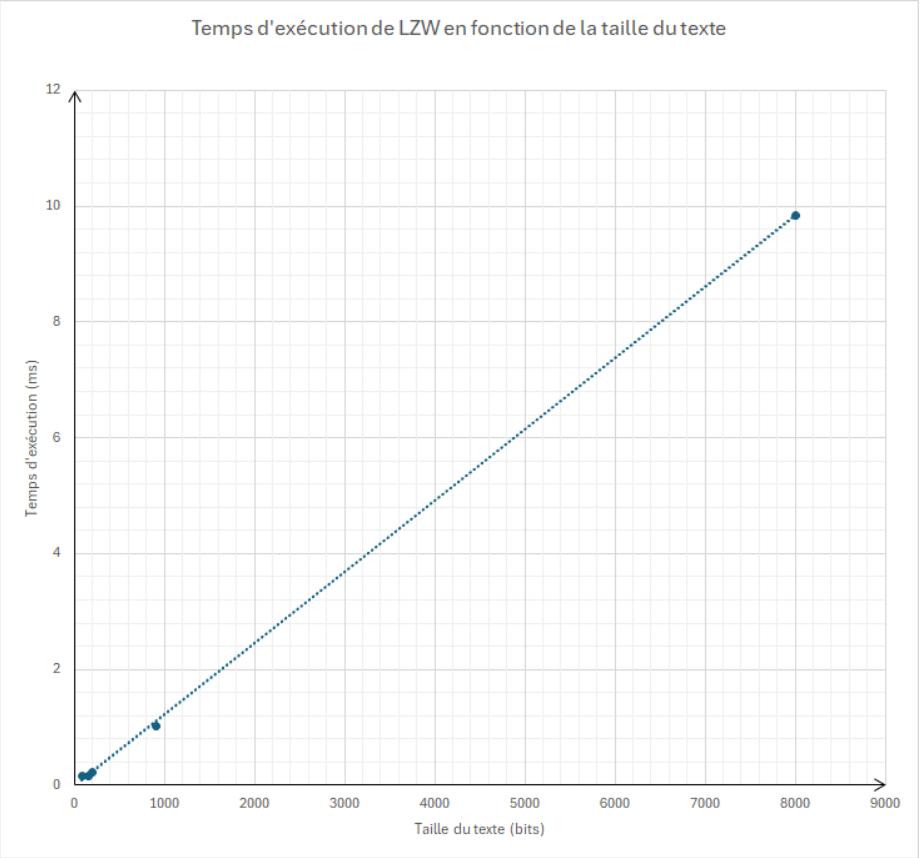
Tableau 2. Performance du codage LZW sur les images

Image	Taux de compression	Temps de compression (s)
1	-0,3321	117,7
2	0,9687	0,2045
3	0,5603	764,9
4	0,1507	1443
5	0.9864	7,737

Les résultats de performances semblent conformes à ce qui était attendu. En effet, les taux de compression sont significatifs (supérieurs ou égaux à 0,10) pour les textes : 1, 2, 3, 5 et pour les images : 2, 3, 4, 5.

Pour les textes, il semble que plus le message est long, plus le temps de compression est long, ce qui est conforme à ce qui est attendu. Cependant, le contenu du message semble également affecter la vitesse de compression. En effet, les temps de compression pour le texte 1 et 2 sont très similaires malgré le fait que le texte 2 est presque deux fois plus long que le texte 1. Ceci s'explique possiblement par le fait que la taille d'un symbole dans le texte 2 est deux fois plus longue que la taille d'un symbole dans le texte 1, ce qui fait en sorte que le nombre de symboles qu'on itère est à peu près le même pour les deux textes. En effet, le

texte 1 contient 2 symboles (ce qui donne 1 bit par symbole) alors que le texte 2 contient 3 symboles (ce qui donne 2 bits par symboles).



Pour les images, il semble que plus la taille de l'image est grande, plus le temps d'exécution augmente. Comme pour les textes, le contenu semble également influencer le temps de compression, plus une image contient d'éléments (cercles, pommes, montagnes, etc), plus la compression est lente. Par exemple, l'image 4 (de taille 6 000 000) nécessite 24 minutes pour compresser alors que l'image 5 (de taille 3 911 280), qui contient moins d'éléments que l'image 5, nécessite 8 secondes.

Question 3: Huffman

Méthodologie: Nous avons utilisé le code fourni sur le GitHub du cours et nous avons utilisé une fonction pour coder les images png en chaînes de caractères afin de réutiliser la même fonction pour les deux types de fichier.

Justification: Dans notre cas, nous avons observé que les fichiers textes étaient de longueur assez courte. Nous avons remarqué que pour des chaînes de caractère courtes l'algorithme LZW donne parfois des taux de compression négatifs. Il est impossible d'obtenir un taux de compression négatif avec l'algorithme de Huffman, le

pire des cas étant de ne pas pouvoir compresser le message, et donc d'obtenir un taux de compression de 0. Dans cette optique, nous supposions que Huffman allait mieux performer que LZW pour les fichiers textes.

Tableau 3. Performance du codage de Huffman sur les textes

Texte	Taux de compression	Temps de compression (ms)
1	0,0000	0,2877
2	0,2338	0,3158
3	0,3578	0,5731
4	0,1879	1,844
5	0,2987	10,20

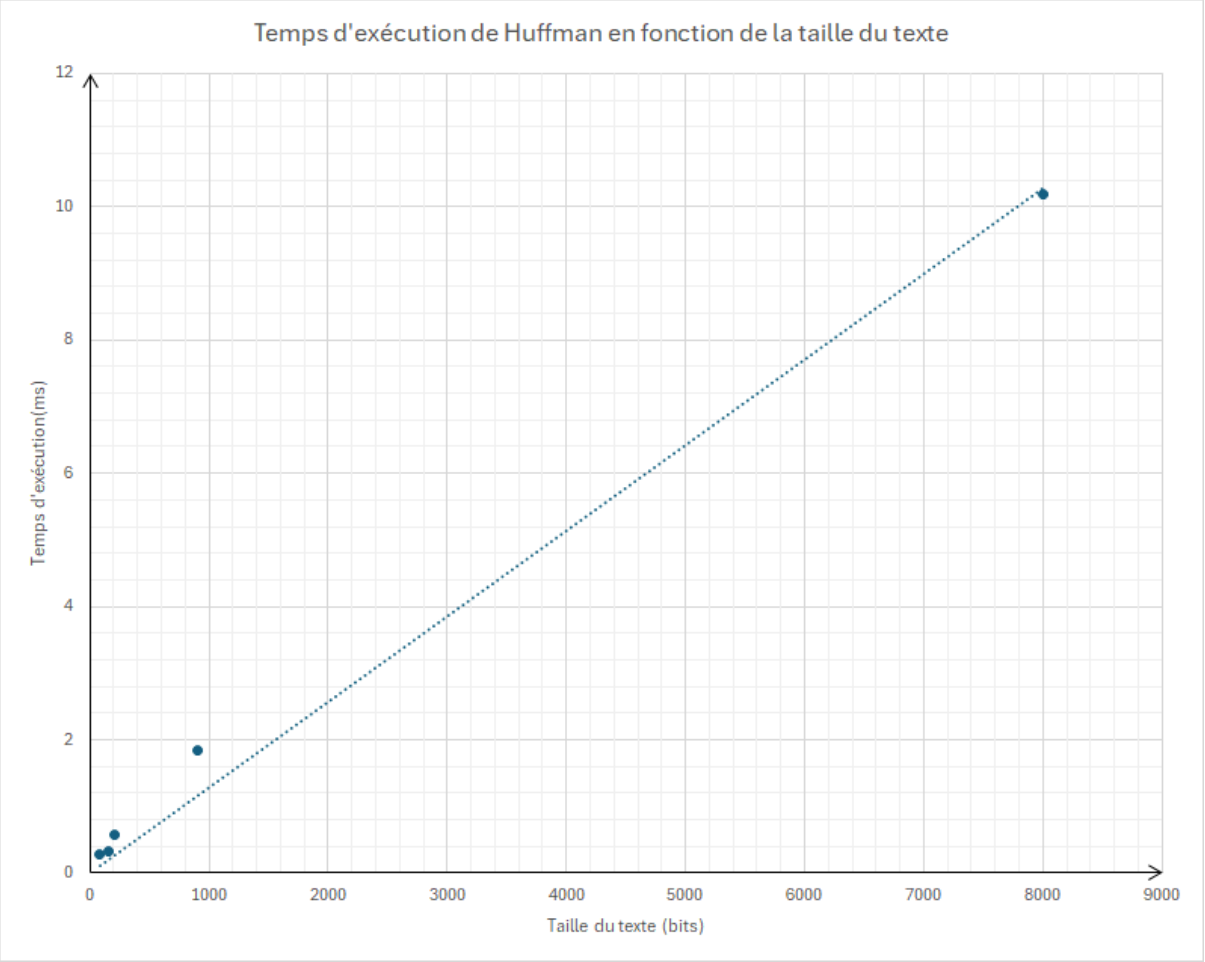
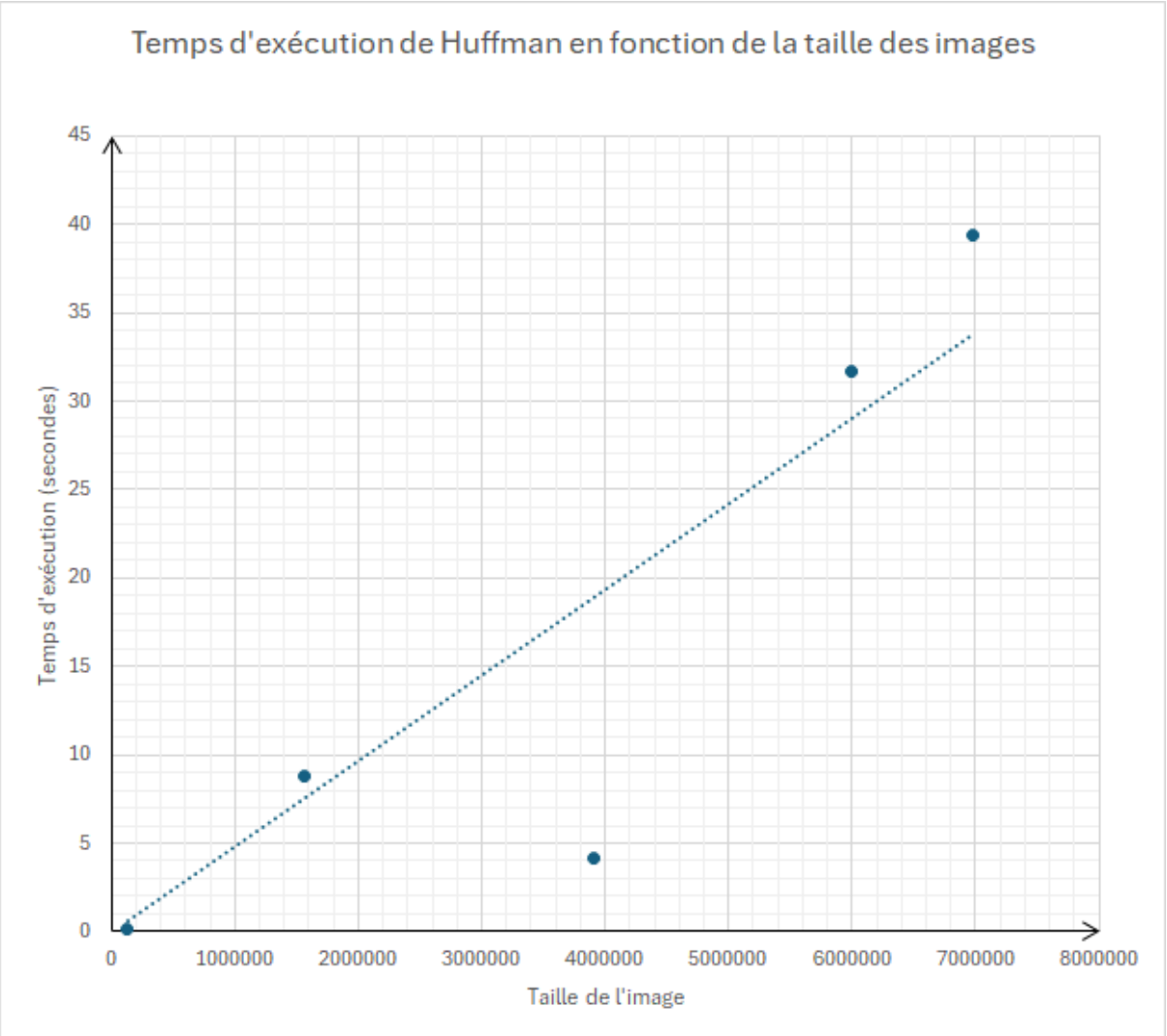


Tableau 4. Performance du codage de Huffman sur les images

Image	Taux de compression	Temps de compression (s)
1	0,0842	8,805
2	0,0000	0,1298
3	0,4637	39,39
4	0,0312	31,73
5	0,6954	4,156



Question 4

On remarque que LZW a mieux performé pour les deux premiers textes, donnant à la fois un meilleur temps d'exécution et un meilleur taux de compression. Ceci n'est pas surprenant sachant que ces deux premiers textes contenaient des sous-chaînes répétitives de 'A' et de 'B'. Pour les trois textes suivants, c'est le codage de Huffman qui à mieux performé en terme de taux de compression alors que LZW a terminé son exécution plus rapidement. À défaut d'avoir de long segment où le même caractère est répété, ces textes sont écrits avec les 26 lettres de l'alphabet plus quelques caractères usuels de la langue latine. Un codage statistique comme Huffman est plus avantageux pour compresser le message grâce à l'analyse fréquentielle qui construit un arbre à quelque branchement, en comparaison avec LZW qui construit dynamiquement un dictionnaire de très grande taille lorsque le message ne contient pas beaucoup de répétition.

Pour les images, on remarque que le taux de compression de LZW est surpassé par Huffman uniquement pour la première, qui est une image où chaque pixel est différent. Cette image est le pire cas pour LZW, qui performe mieux avec les répétitions comme on peut voir avec la deuxième image. Pourtant, on comprend pourquoi c'est Huffman qui est utilisé dans la compression d'image quand on regarde le temps d'exécution. LZW prend jusqu'à 1443 secondes pour compresser une image de haute résolution alors que Huffman prend moins d'une minute pour accomplir la même tâche. Cependant, Huffman rencontre sa limite lorsque l'image contient 2 valeurs ou moins de pixel, comme l'image 2. L'arbre qui est construit dans cette situation ne contient que deux nœuds et aucun gain n'est fait dans la représentation binaire du message.

Les tableaux ci-dessous représentent les gains dans les métriques de performance des codages. Le gain est calculé par la différence entre la valeur obtenue avec Huffman et la valeur obtenue avec LZW. Un gain positif implique que Huffman est plus performant et un gain négatif implique que LZW est plus performant.

Tableau 5. Gains de compression avec Huffman

Numéro du fichier	Gain pour les textes	Gain pour les images
1	-0,1190	0,4052
2	-0,0324	-0,9687
3	0,2500	-0,0966
4	0,1868	-0,1195
5	0,0338	-0,6877

Tableau 6. Gains de temps avec Huffman

Numéro du fichier	Gain pour les textes (ms)	Gain pour les images (s)
1	-0,1368	108,90
2	-0,1604	0,0747
3	-0,3542	725,5
4	-0,821	1411
5	-0,353	3,581