TP1 INF8770

Valentin BOUIS

Benjamin HEINEN

**Question 1**

Hypothèse 1 :

On suppose que moins il y aura de teintes de couleur différentes (exemple : image en noir et blanc), plus l’algorithme de codage par paire d’octets sera efficace. En effet il sera capable de remplacer les répétitions de couleurs par un autre symbole. Dans le cas d’une image en noir et blanc, les répétitions cote a cote de noir et de blanc seraient alors remplacées par un autre symbole, ce qui diminuerait grandement la taille de l’image. Au contraire pour l’algorithme de codage prédictif, l’objectif est d’obtenir le plus de zéro possible dans l’erreur. Ainsi pour une image en noir et blanc on essayerait de transformer les 1 (blanc) en 0 (noir), mais les valeurs initialement à 0 risquent également d’être modifié. On pense donc que le codage par paire d’octets compressera mieux une image avec peu de teintes de couleurs que le codage prédictif.

Hypothèse 2 :

Plus l’image aura de couleurs distinctes qui ne se répètent pas selon un motif, plus le codage prédictif compressera mieux que le codage par paire d’octets. En effet le codage par paire d’octets ne sera pas capable de simplifier des duo de pixels s’il ne trouve pas de motif répété.

Voici un exemple :

Code : 0 1 2 3 4 5

Version paire d’octets : pas de simplification possible (pas de paire identique)

Version prédictif : 0 0 0 0 0 0 dans le cas ou x(n) = x(n-1) - 1

On voit clairement que dans le cas extreme ou aucun motif n’apparait l’algorithme de codage prédictif est bien meilleur que celui par paire d’octets.

Hypothèse 3 :

On pense que le temps d’exécution de l’algorithme de codage par paire d’octets sera plus lent que l’algorithme de codage prédictif lors de la compression. En effet le codage par paire d’octets nécessite un algorithme récursif ainsi que d’établir une map des combinaisons de 2 pixels possibles ainsi que leur nombre d’occurrences. En revanche le codage prédictif ne nécessite seulement que le calcul de l’erreur.

Hypothèse 4 :

La robustesse est un autre critère important pour évaluer la performance d’un algorithme de compression. Celui-ci correspond à « sensibilité de l’algorithme de compression/reconstruction à des petites altérations du code comprimé (erreurs de transmission) » (Telecom ParisTech). Dans le cas de l’algorithme de codage prédictif, si l’erreur de transmission revient à une inversion de 2 symboles, alors seul l’intervalle compris entre les deux symboles sera erroné lors de la décompression. Pour l’algorithme de codage par paire d’octets, une inversion de symboles entrainerait une erreur qui ferait effet boule de neige sur le reste de la chaine à décompresser. Ainsi on pense que l’algorithme de codage prédictif sera plus robuste que l’algorithme de codage par paire d’octets.

Hypothèse 5 :

Le dernier critère serait la taille occupée dans la mémoire lors de la compression. On pense que celui-ci serait plus faible pour l’algorithme de codage prédictif (en admettant qu’on combine celui-ci à l’algorithme de Huffman) car le dictionnaire nécessaire à la compression par paire d’octets sera plus grand puisque il y a plus de duo de symboles (chaque symbole apparait dans 2 duo) que de symboles.

**Question 2**

Test hypothèse 1 :

Pour tester l’efficacité de compression de l’algorithme, nous allons tout simplement regarder la valeur du quotient de compression pour chaque algorithme. Celui-ci est égal à : Taille de l’image initiale/Taille de l’image compressée

On regardera alors ce quotient pour une image en noir et blanc afin de voir si l’algorithme de codage par paire d’octets est bien plus efficace que celui prédictif.

Test hypothèse 2 :

De la meme manière que pour l’hypothèse 1 ce sera le quotient de compression que nous comparerons ici aussi. Cette fois seulement nous utiliserons une image de couleurs avec le moins de motifs apparents possible.

Test hypothèse 3 :

Pour tester le temps d’exécution de chacun des algorithmes, nous allons tout simplement comparer le temps pris par chacun pour compresser une image. Pour cela nous utiliserons le module Time de python. Nous donnerons la même image à traiter aux deux algorithmes. Nous lancerons le timer au commencement de l’algorithme (après l’import de l’image) et l’arrêterons quand celui-ci aura fini de compresser l’image.

Test hypothèse 4 :

Test hypothèse 5 :

**Question 3**

**Question 4**