**ΠΑΝΕΠΙΣΤΗΜΕΙΟ ΠΕΙΡΑΙΩΣ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΑΠΑΛΛΑΚΤΙΚΗ ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΜΑΘΗΜΑ ΣΥΣΤΗΜΑΤΑ ΠΟΛΥΜΕΣΩΝ

*Υλοποιήθηκε απο τον φοιτητή:* ***Βεργιάννη Νικόλαο*** *- Π16170*

# Β ΜΕΡΟΣ ΕΡΓΑΣΙΑΣ

Στόχος της υλοποίησης είναι αλλαγή τυχαίου ψηφίου απο τον αριθμό μητρώου μας *(16170)* με τον αριθμό 5 και η κατασκευή μιας 104x200 εικόνας,η διαίρεση της σε 8x8 macroblocks ώστε να βρεθεί ο μέσος λόγος συμπίεσης της νέας εικόνας που συμπίπτει απο της συνενώση των 2 εικόνων.

**Α εικόνα :**

Για την υλοποίηση της πρώτης εικόνας θα δημιουργήσουμε εναν πίνακα 104x200 με μηδενικά και εφόσον το μήκος των αριθμών είναι 5 και οι στήλες 200 θα έχουμε 40 επαναληψεις. Μετα την τυχαία επιλογή ενός αριθμού απο τον αρχικό πίνακα θα τοποθετηθεί 5 θέσεις μετα το τελευταίο ψηφίο του προηγούμενου αριθμού.

**Β εικόνα :**

Η δεύτερη εικόνα είναι η κατασκευασμένη εικόνα μετα απο την ένωση των macroblocks των DCT συντελεστών. Ακολουθούμε ίδια διαδικασία με την εικόνα Α και έπειτα την χωρίζουμε σε 8x8 macroblocks. Εισάγουμε τους DCT συντελεστές που προέκυψαν απο την εικόνα Α και γίνεται η εισαγωγή.

---

**Κωδικοποίηση Huffman**

Για την κωδικοποίηση Huffman θα χρησιμοποιήσουμε την συνάρτηση Ziz-Zag η οποία ταξινομεί τα στοιχεία του πίνακας ώστε να έχουμε την μορφή [A < C > B]. Η κωδικοποίηση Huffman χρησιμοποιεί τα στοιχεία macroblock που προκύπτουν και αποδίδει ενα κωδικό για κάθε σύμβολο του πίνακα και αποθηκεύει το αποτέλεσμα στο πίνακα output.

---

**Μέσος Λόγος Συμπίεσης**

Για τον υπολογισμό του μέσου λόγου συμπίεσης των δυο εικόνων θα μετρήσουμε τα bits κάθε εικόνας και θα τα αποθηκεύσουμε σε εναν νέο πίνακα βάση των παραπάνω διαδικασιών.

Ο μέσος λόγος υπολογίζεται απο το αποτέλεσμα της διαίρεσης του αθροίσματος των συνολικών λόγων συμπίεσης με το μήκος του.

In [ ]:

```
!pip install huffman library
#import necessary libraries
import huffman
import numpy as np
```

```
Requirement already satisfied: huffman in /usr/local/lib/python3.6/dist-packages (0.1.2)
Requirement already satisfied: library in /usr/local/lib/python3.6/dist-packages (0.0.0)
```

In [ ]:

```
quantization_array = np.array([[16, 11, 10, 16, 24, 40, 51, 61],
                               [12, 12, 14, 19, 26, 58, 60, 55],
                               [14, 13, 16, 24, 40, 57, 69, 56],
                               [14, 17, 22, 29, 51, 87, 80, 62],
                               [18, 22, 37, 56, 68, 109, 103, 77],
                               [24, 35, 55, 64, 81, 104, 113, 92],
                               [49, 64, 78, 87, 103, 121, 120, 101],
                               [72, 92, 95, 98, 112, 100, 103, 99]])

#b = block
def zig_zag_function(b):
    i = 0
    j = 0
    o = 0
    output = np.zeros(64)
```

```
    while ((i<8) and (j<8)):
        if ((i+j) % 2 == 0):
            if (i ==0):
                output[o] = b[i, j]
                if (j == 8):
                    i = i+1
                else:
                    j = j+1
                o = o+1

            elif ((j ==7) and (i < 8)):
                output[o] = b[i, j]
                i = i+1
                o = o+1

            elif ((i > 0) and (j < 7)):
                output[0] = b[i, j]
                i = i-1
                j = j+1
                o = o+1

        else:
            if ((i == 7) and (j <= 7)):
                output[o] = b[i,j]
                j = j+1
                o = o+1

            elif (j == 0):
                output[o] = b[i, j]
                if (i == 7):
                    j = j+1
                else:
                    i = i+1

                o = o+1

            elif ((i < 7) and (j > 0)):
                output[0] = b[i, j]
                j = j-1
                i = i+1
                o = o+1

        if ((i ==7) and (j==7)):
            output[o] = b[i ,j]
            break

    return output


def huffman_encoding(macroblock,output):
    zz = zig_zag_function(macroblock)
    c = huffman.codebook(collections.Counter(zz).items())
    for i in zz:
        output.append(c.get(i))


print("Εισαγωγή αριθμού μητρώου: \n")

id = input()
id_list = []
while(True):
    id_list = id.split()
    if (len(id_list) > 3 or len(id_list)==0 ):
        print('Λάθος εισαγωγή μητρώου')
        id = input()
        continue
    for i in range(len(id_list)):
        if (not id_list[i].isdigit()) or not (len(id_list[i])==5):
            print('Λάθος εισαγωγή Μητρώου')
            id = input()
            continue


    break

print("ID: " + str(id_list))
```

```
print("ID: " + str(id_list))


for i in range(len(id_list)):
    j = random.randint(0,4)
    id_list[i] = id_list[i][:j] + "5" + id_list[i][j+1:]

print("ID μετά την αλλαγή ψηφίου:" + str(id_list))


compressed_ratio = []
#first image 104x200 with zeros
for i in range(100):
    first_image = np.zeros([104,200], dtype=np.float32)
    for j in range(104):
        for k in range(40):
            new_id = random.randint(0, len(id_list)-1)
            for n in range(5):
                first_image[j, k*5 + n] = id_list[new_id][n]

    #second image  104x200 with zeros
    second_image = np.zeros([104,200], dtype=np.float32)

    ##array containing first encoded picture
    huffman1 = []
    #array containing second encoded picture
    huffman2 = []
    for i in range(0, 104, 8):
        for j in range(0, 200, 8):
            huffman_encoding(first_image[i:(i+8), j:(j+8)], huffman1)

    for i in range(0, 104, 8):
        for j in range(0, 200, 8):
            second_image[i:(i+8),j:(j+8)] = cv2.dct(first_image[i:(i+8),j:(j+8)])
            second_image[i:(i+8),j:(j+8)] = np.ceil(np.divide(second_image[i:(i+8),j:(j+8)], quanti:
ation_array))
            huffman_encoding(second_image[i:(i+8),j:(j+8)],huffman2)



    first_image_size = len(''.join(huffman1))
    second_image_size = len(''.join(huffman2))

    compressed_ratio.append(first_image_size/second_image_size)

final_commpresed_ratio = sum(compressed_ratio)/len(compressed_ratio)
print("Average compression ratio: " + str(final_commpresed_ratio))
```

```
Εισαγωγή αριθμού μητρώου:

16170
ID: ['16170']
ID μετά την αλλαγή ψηφίου:['56170']
Average compression ratio: 2.145833333333336
```