ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής

Εργασία Μαθήματος «***Θεωρία πληροφοριών και κωδίκων***»

Αριθμός εργασίας – Τίτλος εργασίας	***Τελική εργασία***
Ονόματα φοιτητή	**Κωνσταντίνος Χριστογεώργος**
Αρ. Μητρώου	**Not public information**
Ημερομηνία παράδοσης	**23/07/2021**

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1 Περιγραφή κώδικα της λύσης..... 3

2 Ενδεικτικό παράδειγμα από την λειτουργία του προγράμματος..... 8

3 Βιβλιογραφικές Πηγές.....10

1 Περιγραφή κώδικα της λύσης

Το .sagews αρχείο αποτελείται από 3 συναρτήσεις και τον υπόλοιπο "Driver" κώδικα για να υλοποιήσει την λύση για το αρχικό πρόβλημα.

- Συνάρτηση **ProduceCode(n,k)**

```
def ProduceCode(n,k):
    while(True):
        G1 = random_matrix(ZZ, k, k, x=0, y=2)
        S = random_matrix(ZZ, n, k, x=0, y=2)
        P = identity_matrix(k, k)
        G = matrix(GF(2), S * G1 * P)
        C = LinearCode(G)
        if(C.generator_matrix().nrows() != n):
            continue
        else:
            break
    return C
```

Η συνάρτηση αυτή παράγει και επιστρέφει τον γραμμικο κώδικα. Δέχεται ως ορίσματα 2 ακέραιους αριθμούς και κατόπιν δημιουργεί 3 τυχαίους πίνακες G1, S και P.

Ο πίνακας G1 είναι διαστάσεων (k,k), ο S είναι διαστάσεων (n,k) και ο P είναι μοναδιαίος διαστάσεως (k,k). Ο πολλαπλασιασμός των 3 αυτών πινάκων παράγει τον πίνακα G (n,k) διαστάσεων, ο οποίος χρησιμοποιείται για την παραγωγή του γραμμικού κώδικα. Εφόσον ο κώδικας είναι n γραμμών (κάποιες φορές δεν ήταν και για αυτό προστέθηκε αυτό το σημείο στον κώδικα), αυτός επιστρέφεται.

- Συνάρτηση **binaryToDecimal(binary)**

```
def binaryToDecimal(binary):
    decimal, i, n = 0, 0, 0
    while (binary != 0):
        dec = binary % 10
        decimal = decimal + dec * pow(2, i)
        binary = binary // 10
        i += 1
    return decimal
```

Η συνάρτηση αυτή δέχεται ως όρισμα έναν binary αριθμο και τον μετατρέπει σε δεκαδικό.

- Συνάρτηση **binary_representation_of_message(message, encoding_length)**

```
def binary_representation_of_message(message, encoding_length):
    asciivalues = [ord(c) for c in message]
    print("ASCII VALUES OF ORIGINAL MESSAGE: ", asciivalues)

    binaryvalues = [("0" * 8 + bin(c)[2:])[-8:] for c in asciivalues]
    print("BINARY VALUES OF ORIGINAL MESSAGE: ", binaryvalues)

    binarystring = "".join(binaryvalues)
    print("CONCATENATED BINARY VALUES INTO A STRING: ", binarystring)

    original_length = len(binarystring)
    print("LENGTH OF ORIGINAL MESSAGE: ",len(binarystring))
    length = len(binarystring) + 32
    print("LENGTH WITH 32 BITS ADDED: ", length)
    total_length = 0
    iterator = 1
    while True:
        if length % encoding_length == 0:
            total_length = length
            break
        else:
            if encoding_length*iterator > length:
                total_length = encoding_length*iterator
                break
            else:
                iterator = iterator + 1
                continue


    print("LENGTH WITH 32 BITS ADDED AND WITH NEEDED BITS FOR ENCODING ADDED: ", total_length)

    binary_length = ("0" * 8 + bin(len(binarystring))[2:])[-8:]
    print("ORIGINAL MESSAGE'S LENGTH IN BINARY: ", binary_length)

    if total_length - length > 0:
        binarystring = "0"*(total_length - length) + binarystring
    print("MESSAGE WITH", total_length, "-", length,"=",total_length - length, " 0 BITS ADDED ON THE LEFT: ", binarystring)

    binarystring = 4*(str(binary_length)) + binarystring
    print("MESSAGE WITH", total_length - length, "0 BITS ADDED ON THE LEFT AND WITH 4 TIMES THE ORIGINAL MESSAGE'S BINARY LENGTH ADDED: ", binarystring)
    print("")
    return binarystring,total_length,encoding_length,original_length
```

Η συνάρτηση αυτή δέχεται ως ορίσματα το μήνυμα για αποκωδικοποίηση και το μέγεθος του εκάστοτε υπο-μηνύματος. Αρχικά μετατρέπει τον κάθε χαρακτήρα του μηνύματος στην ascii τιμή του και αυτήν σε binary. Κατόπιν, η σύμβαση για το τέλος του μηνύματος είναι να προστεθεί το binary μέγεθος του μηνύματος 4 φορές στα αριστερά του binary μηνύματος.

Έστω k ακέραιος που ανήκει στο N*.
Πριν την κωδικοποίηση, για να είναι σίγουρο πως θα κωδικοποιούνται k-δες από χαρακτήρες του μηνύματος κάθε φορά, προσθέτουμε πριν το μέγεθος όσα μηδενικά bits χρειάζονται έτσι ώστε το μέγεθος σε bit του αρχικού μηνύματος + 32 bit(4 byte από το μέγεθος του μηνύματος) + τα αναγκαία μηδενικά bits να διαιρούνται ακριβώς από το k.
Έτσι, αφότου έχει παραχθεί το τελικό string, επιστρέφεται για κωδικοποίηση.

- **Driver Code**

  Αρχικά, ορίζουμε το μήνυμα προς κωδικοποίηση και τα n,k διαστάσεις του πίνακα γεννήτορα και τα bit του υπο-μηνύματος. Έπειτα, εφόσον έχουμε καλέσει την **binary_representation_of_message()** και έχουμε δημιουργήσει τους γραμμικούς κώδικες, στέλνουμε για κωδικοποίηση ανά n bits του μηνύματος από τα δεξιά προς τα αριστερά.
  Οι κώδικες δημιουργούνται με μέριμνα η ελάχιστη απόσταση να είναι μεγαλύτερη από 2, έτσι ώστε να μπορεί να εντοπιστεί και να διορθωθεί τουλάχιστον 1 bit που ίσως να είναι λανθασμένο λόγω θορύβου.

```
print("FOR C1 CODE")
while(True):
    C1 = ProduceCode(n,k)
    print(C1.generator_matrix())
    print("MINIMUM DISTANCE OF C1: ", C1.minimum_distance())
    if C1.minimum_distance() >= 2:
        break
    else:
        continue
print("")

print("FOR C2 CODE")
while(True):
    C2 = ProduceCode(k,k+3)
    print(C2.generator_matrix())
    print("MINIMUM DISTANCE OF C2: ", C2.minimum_distance())
    if C2.minimum_distance() >= 2:
        break
    else:
        continue
print("")
```

Πρώτα, κωδικοποιείται με τον κώδικα C1(n,k) το εκάστοτε υπο-μήνυμα και έπειτα με τον κώδικαC2(k,k+3). Τότε, με πιθανότητα ¼ ένας bit θορύβου προστίθεται σε τυχαία θέση του αποκωδικοποιημένου υπο-μηνύματος. Μετά, αποκωδικοποιούμε αρχικό με τον C2 και μετρά με τον C1 για να πάρουμε το πλήρως αποκωδικοποιημένο υπο-μήνυμα. Αυτό επαναλαμβάνεται για όλα τα bits του μηνύματος.

```
for i in range(0, total_length, encoding_length):
    print("ORIGINAL MESSAGE BEFORE ENCODING: ", string_to_code[i:i + encoding_length])
    temp = []
    temp = list(string_to_code[i:i + encoding_length])
    for c in range(len(temp)):
        temp[c] = int(temp[c])
    message = vector(temp)

    print("MESSAGE AFTER ENCODING WITH C1")
    C1.encode(message)
    c1_encoded = C1.encode(message)
    print("MESSAGE AFTER ENCODING WITH C2")
    C2.encode(c1_encoded)
    c2_encoded = C2.encode(c1_encoded)

    while(True):
        random_iteration = random.randint(0,4)
        random_position = random.randint(0,k+2)
        if(random_iteration == 1):
            error = zero_vector(GF(2), k+3)
            print("ADDING RANDOM ERROR NOW")
            for i in range(k+3):
                if random_position == i:
                    error[i] = 1
                    break
            c2_encoded = c2_encoded + error
        break

    print("MESSAGE AFTER ENCODING WITH C2 + POSSIBLE ERROR", c2_encoded)
    print("DECODED MESSAGE FROM C2")
    c2_decoded = C2.decode_to_message(c2_encoded)
    print(c2_decoded)

    print("DECODED MESSAGE FROM C1")
    c1_decoded = C1.decode_to_message(c2_decoded)
    print(c1_decoded)
    decoded_list.append(c1_decoded)
    print("")
```

Τα αποκωδικοποιημένα μηνύματα αποθηκεύονται σε μια λίστα.
Μέσω της λίστας, κρατάμε τα bits που είναι δεξιότερα από τα 32 bits ελέγχου + όσα προσδέθηκαν για να μπορεί να χωριστεί σε k-δες bits.
Τότε, αυτά τα bits χωρίζονται σε 8-δες και μετατρέπονται σε δεκαδικοί αριθμοί(ascii) και αυτοί στους αντίστοιχους χαρακτήρες των ascii τιμών τους.

```
for i in range(len(decoded_list)):
    decoded_list[i] = list(decoded_list[i])

decoded_string = ""
for sublist in decoded_list:
    for item in sublist:
        decoded_string += str(item)
    decoded_string += ""


decoded_string =  decoded_string[total_length - original_length:]
print("FULLY DECODED BINARY STRING: ",decoded_string[total_length - original_length:])

final_string = ""
for c in range(0, len(decoded_string), 8):
    final_string = final_string + "".join(chr(binaryToDecimal(int(decoded_string[c:c+8]))))

print(final_string)
```

2 Ενδεικτικό παράδειγμα από την λειτουργία του προγράμματος

- Μήνυμα για κωδικοποίηση το string: **coron2as182+**
- **n = 4, k =7**

Παρακάτω φαίνεται οι ascii binary τιμές του μηνύματος σε ένα string το οποίο κωδικοποιείται σε 4δες. Εδώ είναι 96 bit αρχικό μήνυμα + 32 = 128 που διαιρείται με το 4 αρά δεν χρειάζονται extra bits.

```
MESSAGE:  coron2as182+
ASCII VALUES OF ORIGINAL MESSAGE:  [99, 111, 114, 111, 110, 50, 97, 115, 49, 56, 50, 43]
BINARY VALUES OF ORIGINAL MESSAGE:  ['01100011', '01101111', '01110010', '01101111', '01101110', '00110010', '01100001', '01110011', '00110001', '00111000', '00110010', '00101011']
CONCATENATED BINARY VALUES INTO A STRING:  011000110110111101110010011011101110001100100110000101110011001100010011100000110010001011


LENGTH OF ORIGINAL MESSAGE:  96
LENGTH WITH 32 BITS ADDED:  128
LENGTH WITH 32 BITS ADDED AND WITH NEEDED BITS FOR ENCODING ADDED:  128
ORIGINAL MESSAGE'S LENGTH IN BINARY:  01100000
MESSAGE WITH 128 - 128 = 0  0 BITS ADDED ON THE LEFT:  01100011011011110111001001101111011011100011001001100001011100110011000100111000001100100010111
MESSAGE WITH 0 0 BITS ADDED ON THE LEFT AND WITH 4 TIMES THE ORIGINAL MESSAGE'S BINARY LENGTH ADDED:
01100000011000000110000001100000011000011011011110111001001101111011011100011001001100001011100110011000100111000001100100010111
```

Οι πίνακες γεννήτορες των C1,C2 με ελάχιστη απόσταση >= 2.

```
FOR C1 CODE
[1 1 0 1 0 1 0]
[1 1 1 1 1 1 1]
[1 1 0 1 1 0 1]
[1 0 0 0 1 1 0]
MINIMUM DISTANCE OF C1:  2

FOR C2 CODE
[1 1 0 1 0 1 0 0 1 1]
[1 0 0 0 0 0 1 0 1 1]
[0 1 0 1 1 1 0 0 0 1]
[0 0 0 1 1 0 1 0 0 1]
[0 1 1 0 1 0 1 0 1 1]
[0 0 1 0 1 1 1 0 1 1]
[0 1 0 1 1 1 0 0 1 1]
MINIMUM DISTANCE OF C2:  1
[1 1 1 1 1 1 1 0 1 0]
[0 1 1 1 1 0 1 0 0 1]
[1 1 1 1 0 1 0 1 0 0]
[0 0 1 1 0 0 0 1 0 0]
[1 1 1 1 0 1 1 0 0 1]
[1 0 0 1 1 1 0 0 1 0]
[1 0 0 1 0 1 0 1 1 1]
MINIMUM DISTANCE OF C2:  1
[1 0 1 1 0 1 0 1 1 1]
[0 0 0 0 1 1 1 1 0 1]
[1 1 1 0 1 0 1 1 1 1]
[0 1 0 0 1 1 1 1 1 1]
[0 1 1 1 0 1 0 0 0 1]
[1 1 1 0 0 1 1 0 1 1]
[0 0 0 1 0 0 1 1 0 1]
MINIMUM DISTANCE OF C2:  2
```

- Και κωδικοποιούμε/αποκωδικοποιούμε με τους κώδικες ανά 4-δες.

```
ORIGINAL MESSAGE BEFORE ENCODING:  0110
MESSAGE AFTER ENCODING WITH C1
(0, 0, 1, 0, 0, 1, 0)
MESSAGE AFTER ENCODING WITH C2
(0, 0, 0, 0, 1, 1, 0, 1, 0, 0)
MESSAGE AFTER ENCODING WITH C2 + POSSIBLE ERROR (0, 0, 0, 0, 1, 1, 0, 1, 0, 0)
DECODED MESSAGE FROM C2
(0, 0, 1, 0, 0, 1, 0)
DECODED MESSAGE FROM C1
(0, 1, 1, 0)
```

- Σε αυτή την επανάληψη εισάγεται θόρυβος που διορθώνεται σωστά.

```
ORIGINAL MESSAGE BEFORE ENCODING:  0000
MESSAGE AFTER ENCODING WITH C1
(0, 0, 0, 0, 0, 0, 0)
MESSAGE AFTER ENCODING WITH C2
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
ADDING RANDOM ERROR NOW
MESSAGE AFTER ENCODING WITH C2 + POSSIBLE ERROR (0, 1, 0, 0, 0, 0, 0, 0, 0, 0)
DECODED MESSAGE FROM C2
(0, 0, 0, 0, 0, 0, 0)
DECODED MESSAGE FROM C1
(0, 0, 0, 0)
```

Εδώ δεν διορθώνεται σωστά.

```
ORIGINAL MESSAGE BEFORE ENCODING:  0001
MESSAGE AFTER ENCODING WITH C1
(1, 0, 0, 0, 1, 1, 0)
MESSAGE AFTER ENCODING WITH C2
(0, 0, 1, 0, 0, 1, 1, 1, 0, 1)
ADDING RANDOM ERROR NOW
MESSAGE AFTER ENCODING WITH C2 + POSSIBLE ERROR (0, 0, 1, 0, 1, 1, 1, 1, 0, 1)
DECODED MESSAGE FROM C2
(0, 1, 0, 0, 0, 0, 0)
DECODED MESSAGE FROM C1
(0, 0, 0, 0)
```

- Και το πλήρως αποκωδικοποιημένο μήνυμα.

```
FULLY DECODED BINARY STRING:  011011100011001001100000011100110011000100111000001100100010101 1
coron2`s182+
```

Ελαφρώς διαφορετικό από το αρχικό λόγω του θορύβου.

3 Βιβλιογραφικές Πηγές

https://www.youtube.com/watch?v=kO6UlCY6idg&t=550s

https://www.youtube.com/watch?v=as_mNSx6OG8

https://www.youtube.com/watch?v=VasEzbjCDJ4&t=4s

https://doc.sagemath.org/html/en/reference/coding/sage/coding/linear_code.html