МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ   
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”



КОД ХЕММІНГА

**Звіт до лабораторної роботи №6**

**з курсу “Теорія інформації та кодування”**

Варіант 36

Виконав:  
ст. гр. ІР-21

Касараба Володимир  
  
Прийняв:  
Стахів Р. І.

Львів – 2021

**Мета роботи**: вивчення характеристики і принципу побудови коду Хеммінга.

**Завдання**

1.За вказівкою викладача з таблицi 1, згідно з номером списку в журналі групи, вибрати десяткову комбінацію, яку необхідно перевести в комбінацію коду Хеммінга і описати основні характеристики.

2. Розробити алгоритм програми перетворення вказаної комбінації простого двійкового коду в комбінацію коду Грея.

3. Розробити алгоритм програми декодування вказаної комбінації коду Грея в комбінацію простого двійкового коду.

|  |  |
| --- | --- |
| Варіант | Десяткова комбінація  *N10* |
| 36 | 798 |

**Послідовність виконання роботи**

1. Задану десяткову (табл. 1) комбінацію перетворити в комбінацію простого двійкового коду. 2. Для розроблених алгоритмів перетворення комбінації простого двійкового коду в комбінацію коду Хеммінга та її декодування набрати програми мовою Python і відладити її.

3. Виконати кодування та декодування вказаної десяткової комбінації, використовуючи розроблені програми і порівняти одержаний код з виконаними попередньо розрахунками.

4. Ввести помилку в 5-тому розряді коду Хеммінга і перевірити можливість виявлення та виправлення помилки розробленою програмою.

5. Роздрукувати одержані результати кодування/декодування, а також розроблені програми.

6. Зробити висновки стосовно результатів виконаної роботи .

**Виконання**

Задана десяткова комбінація: 798

Комбінація двійкового коду: 1100011110

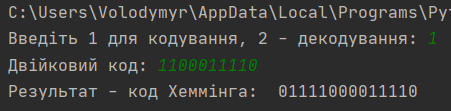
**Програма кодування/декодування коду**

**Хеммінга на мові Python:**

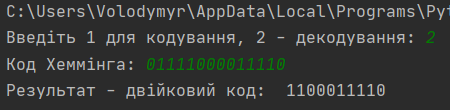
from functools import reduce  
  
def find\_redundant\_bits(bits):  
 length = len(bits)  
 redundant\_bits = []  
 for i in range(length):  
 if (2 \*\* i >= length + i + 1):  
 return i+1 , redundant\_bits  
 redundant\_bits.append(2 \*\* i)  
  
def bin\_to\_hamming(bits):  
 hamming = []  
 length = len(bits)  
 no\_redun\_bits = find\_redundant\_bits(bits)[0]  
 redundant\_bits = find\_redundant\_bits(bits)[1]  
 bits\_iter = iter(bits)  
 for i in range(1, length + no\_redun\_bits):  
 if i in redundant\_bits:  
 hamming.insert(i-1,0)  
 else:  
 hamming.append(next(bits\_iter))  
  
 for bit\_pos in redundant\_bits:  
 sum = 0  
 arr\_index = bit\_pos-1  
 while arr\_index < len(hamming):  
 for bit\_pos\_temp in range(bit\_pos):  
 if arr\_index + bit\_pos\_temp < len(hamming):  
 sum += hamming[arr\_index + bit\_pos\_temp]  
 else:  
 break  
 arr\_index += bit\_pos \* 2  
 hamming[bit\_pos-1] = sum % 2  
 return ''.join([str(i) for i in hamming[0:]])  
  
def find\_broken\_bit\_pos(hamming):  
 return reduce(  
 lambda x, y: x ^ y,  
 [i for (i,b) in enumerate(hamming) if b]  
 )  
  
def hamming\_to\_bin(hamming):  
 hamming.insert(0, sum(hamming) % 2)  
 broken\_bit = find\_broken\_bit\_pos(hamming)  
 redundant\_bits = []  
 if broken\_bit:  
 raise ValueError(f'Біт {broken\_bit} спотворений')  
 for i in range(len(hamming)):  
 if (2 \*\* i >= len(hamming)):  
 break  
 else:  
 redundant\_bits.append(2 \*\* i)  
 for x in redundant\_bits[::-1]:  
 del hamming[x]  
 return ''.join([str(i) for i in hamming[1:]])  
  
choice = input('Введіть 1 для кодування, 2 - декодування: ')  
if choice == "1":  
 binary = input('Двійковий код: ')  
 bits = [int(i) for i in binary]  
 print('Результат - код Хеммінга: ', str(bin\_to\_hamming(bits)))  
if choice == "2":  
 hamming\_code = input('Код Хеммінга: ')  
 bits = [int(i) for i in hamming\_code]  
 print('Результат - двійковий код: ', str(hamming\_to\_bin(bits)))

**Результат виконання програми:**

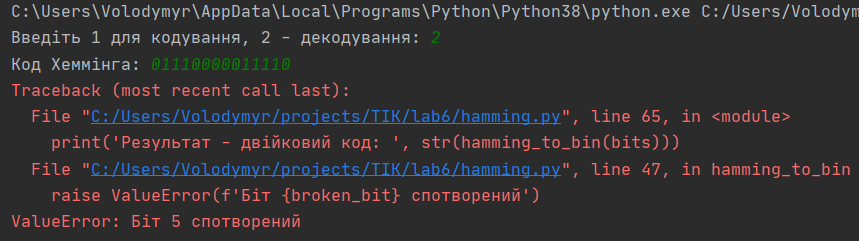
1. Закодуємо вказане двійнове число у код Хеммінга



1. Декодуємо вказаний код Хеммінга у двійкове число



1. Навмисно змінемо 5й біт коду Хеммінга, щоб переконатись, що програма видасть відповідне повідомлення про спотворення сигналу.



**Аналіз:**

Як бачимо, програма кодування та декодування працює правильно. Результат виконання – це правильно обраховані значення; перевірка на спотворення сигналу працює коректно.

**Висновок:**

На цій лабораторній роботі я вивчив характеристики і принципи побудови коду Хеммінга, написав програму мовою Python для кодування та декодування двійкових чисел. Згідно варіанту, для кодування я використав десяткове число 798, двійква форма якого 11000111110.

Реалізуючи програму кодування/декодування для коду Хеммінга, я використав спрощений код Хеммінга з мінімальною відстанню dmin =3. Закодувавши двійкове число 1100011110, я отримав код Хеммінга 01111000011110. Зробивши декодування, я отримав початкове двійкове число. Після я зробив перевіку обрахунків вручну, та переконався, що програма працює правильно. Також, змінивши 5й біт, я вдостовірився, що програма видає повідомлення про спотворення сигналу, якщо передати неправильний код Хеммінга.

Код програми та скріншоти результатів я заніс у звіт.