Лабораторная работа № 5

Дискретная математика для программистов

Отт Арсений 2ПМ

Вариант 1

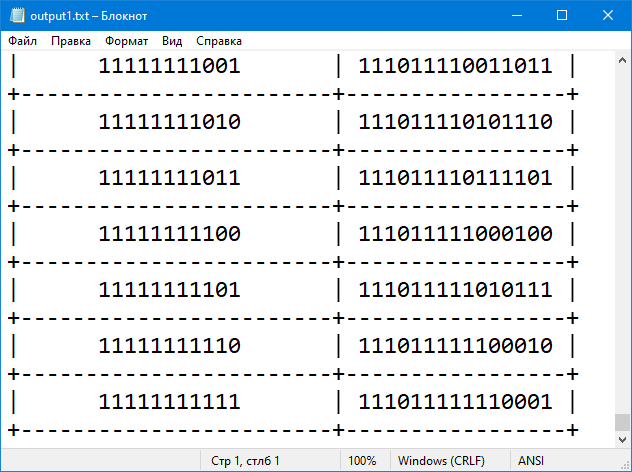
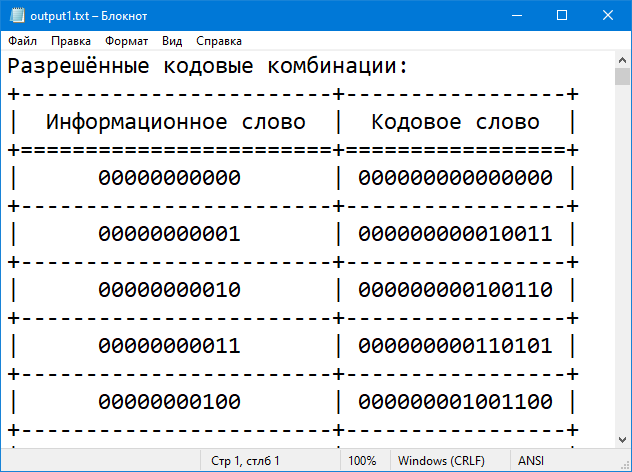
Пусть – общее число элементов, – число информационных элементов, – число избыточных элементов,  
.

Порождающий многочлен: символическая запись – ; двоичная запись –

1. **Найдите все кодовые слова заданного кода и постройте таблицу с разрешенными кодовыми комбинациями и с минимальными расстояниями Хемминга для разрешенных комбинаций; в отчете привести фрагменты таблиц**

Согласно параметрам заданного кода, возможно, закодировать двоичные числа от нуля до . Кодовые слова будем находить умножением векторов-строк размера на порождающую матрицу размера

Порождающая матрица представляет собой прямоугольную числовую матрицу, у которой первая строка – дополненный нулями справа порождающий многочлен, вторая – первая строка, смещённая на столбец вправо, и так далее. Последняя, двадцать шестая строка, представляет собой порождающий многочлен, дополненный нулями слева



**import** sys  
**import** numpy **as** np  
**import** itertools  
  
np.set\_printoptions(threshold=sys.maxsize)  
  
  
**def** hammingDist(str1, str2, fillchar=**'-'**):  
 **return** sum([ch1 != ch2 **for** (ch1, ch2) **in** itertools.zip\_longest(str1, str2, fillvalue=fillchar)])  
  
  
g = [int(i) **for** i **in "10011"**] *# порождающий многочлен*n = 15 *# общее число элементов*m = 11 *# число информационных элементов*length = len(g)  
**for** i **in** range(n - length):  
 g.append(0)  
G0 = [] *# временная матрица***for** i **in** range(m):  
 G0.append(np.roll(g, i))  
G = np.array(G0) *# порождающая матрица*codewords\_table = np.array(**"00000000000"**)  
length = pow(2, m)  
**for** i **in** range(1, length):  
 d = np.array([int(i) **for** i **in** np.binary\_repr(i, m)])  
 codewords\_table = np.append(codewords\_table, **''**.join(map(str, np.mod(d.dot(G), 2))))  
  
length = len(codewords\_table)  
hamming\_distances\_table = np.zeros((length, length), dtype=int)  
**with** open(**'output2.txt'**, **'w'**) **as** file:  
 file.write(**"Таблица кодовых расстояний:\n"**)  
 file.write(**"\n"**)  
 **for** i **in** range(0, length):  
 **for** j **in** range(i + 1, length):  
 hamming\_distances\_table[i][j] = hammingDist(codewords\_table[i], codewords\_table[j])  
 hamming\_distances\_table[j][i] = hamming\_distances\_table[i][j]  
 file.write(str(hamming\_distances\_table[i]))  
 file.write(**"\n"**)  
 file.write(**"\n"**)  
 file.write(**"Минимальное кодовое расстояние:\n"**)  
 file.write(str(np.amin(np.where(hamming\_distances\_table == 0, 100, hamming\_distances\_table))))  
 file.close()

1. **Определите характеристики заданного кода в режиме исправления ошибок:**
   1. **Определите кратность гарантированно исправляемых кодом ошибок**

*Утверждение*. Код исправляет в сообщении ошибок, если кодовое расстояние не меньше, чем , то есть

– кратность гарантированно исправляемых кодом ошибок

* 1. **Найдите число различных векторов ошибок, которые код может исправить**

*Утверждение*. Общее число различных векторов ошибок, которые может исправить циклический код, равно

– число различных векторов ошибок, которые может исправить данный код

* 1. **Для одного из векторов ошибок, исправляемых кодом, найдите соответствующий этому вектору синдром. Найдите несколько из возможных векторов ошибок, при декодировании которых получается тот же синдром, и, следовательно, происходит ошибочное декодирование**

– вектор ошибки:

Пусть вектор ошибки , то есть ошибка произошла в -ом бите

Рассмотрим следующее кодовое слово:

После передачи по каналу связи оно примет вид:

Найдём синдром ошибки

Найдём



|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Найдём

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Таким образом,

1. **Определите возможности заданного кода в режиме обнаружения ошибок:**
   1. **Определите кратность гарантированно обнаруживаемых кодом ошибок**

*Утверждение*. Код обнаруживает в сообщении ошибок, если кодовое расстояние не меньше, чем , то есть

– кратность гарантированно обнаруживаемых кодом ошибок

* 1. **Найдите векторы ошибок, которые не могут быть обнаружены заданным кодом**

Так как слова любого линейного кода обладают свойством замкнутости по отношению к операции сложения, то есть сумма двух и более кодовых слов тоже является кодовым словом, то векторы ошибок, совпадающие с кодовыми словами, не могут быть обнаружены декодером