Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт
по практической работе №4 «Определение параметров линейного кода»

Выполнил:
студент группы ИП-014
Бессонов А.О.

Работу проверил:
старший преподаватель
Дементьева К.И.

Новосибирск 2024 г.

Постановка задачи

Цель работы: Изучение свойств линейного корректирующего кода

Язык программирования: C, C++, C#, Python

Результат: программа, тестовые примеры, отчет.

Задание:

1. Порождающая матрица записана в текстовом файле. Файл имеет следующий формат: в первой строке через пробел записаны два натуральных числа n (количество строк матрицы) и m (количество столбцов), в следующих n строках записаны через пробел по m нулей и единиц. Файл можно генерировать случайно.

Пример файла

```
3 5
1 0 1 1 1
0 1 0 1 0
0 0 1 1 1
```

2. По заданной порождающей матрице определить характеристики линейного кода: размерность кода, количество кодовых слов, минимальное кодовое расстояние. Использовать 5 различных файлов.

Ход работы

Файлы с порождающими матрицами генерируются случайным образом в программе, с условием того, что число столбцов не меньше числа строк, а так же с условием того, что строки в матрице не повторяются.

Результаты работы программы:

```
Filename : matrix_file_1.txt
Generator matrix (n=2, m=10)
1 1 0 1 1 1 0 0 0 0
1 1 1 0 1 0 0 0 1 0

Code dimension = 2
Number of code words = 4
Minimum code distance = 4
```

```
Filename : matrix_file_2.txt
Generator matrix (n=10, m=10)
0 0 0 1 1 1 1 0 1 1
1 1 1 0 0 0 1 1 1 0
1 1 0 1 1 0 0 1 0 1
1 0 1 1 0 0 1 1 1 1
1 0 1 0 1 1 0 1 1 1
0 0 1 0 1 1 1 1 1 0
1 0 1 0 0 1 0 1 0 0
0 1 0 0 1 0 1 0 1 0
1 1 0 1 1 1 1 1 0 0
1 0 0 0 1 0 0 0 0 1

Code dimension = 10
Number of code words = 1024
Minimum code distance = 3
```

```
Filename : matrix_file_3.txt
Generator matrix (n=6, m=10)
1 1 0 0 0 1 1 1 0 1
1 0 0 1 1 0 1 0 1 0
1 0 0 1 0 1 0 1 0 0
1 1 0 0 0 1 0 1 0 0
1 0 0 1 1 0 1 0 0 0
0 0 0 0 1 1 0 1 1 1

Code dimension = 6
Number of code words = 64
Minimum code distance = 1
```

```
Filename : matrix_file_4.txt
Generator matrix (n=7, m=9)
1 0 1 0 0 1 1 0 0
1 0 0 0 0 1 1 0 0
1 0 0 1 0 1 1 0 1
1 0 0 1 0 0 0 1 1
1 0 0 1 1 1 0 0 0
0 0 1 1 1 0 0 1 1
0 0 1 1 0 0 0 1 1

Code dimension = 7
Number of code words = 128
Minimum code distance = 1
```

```
Filename : matrix_file_5.txt
Generator matrix (n=2, m=5)
0 1 1 0 0
1 0 1 1 0

Code dimension = 2
Number of code words = 4
Minimum code distance = 3
```

Выводы

В практической работе было реализовано нахождение свойств линейного корректирующего кода по файлу с порождающей матрицей. Можно заметить, что все свойства напрямую зависят от порождающей матрицы.

Размерность кода – это количество строк порождающей матрицы. А количество кодовых слов – это степень двойки от количества строк порождающей матрицы.

То есть для влияния на эти свойства надо изменять количество строк порождающей матрицы.

Минимальное кодовое расстояние— это минимальное различие между всеми парами строк порождающей матрицы.

Значит, чтобы увеличить или уменьшить минимальное кодовое расстояние надо изменять значения порождающей матрицы. Для обеспечения наилучшей надежности линейного корректирующего кода минимальное кодовое расстояние должно быть наибольшим из возможных, так ошибки при передаче сообщение будут исправляться с большей вероятности, что обеспечит более высокую надежность.

Соответственно по порождающей матрице можно сразу судить о свойствах линейного корректирующего кода.

Код программы

```
#include <iostream>
#include <fstream>
#include <random>
#include <string>
#include <vector>

using namespace std;

void generate_file(string filename) {
	ofstream output(filename);

	random_device rd;
	mt19937 gen(rd());

	uniform_int_distribution<int> n_distribution(1, 10);
	int n = n_distribution(gen);

	uniform_int_distribution<int> m_distribution(n, 10);
	int m = m_distribution(gen);

	output << n << " " << m << "\n";

	uniform_int_distribution<int> bool_distribution(0, 1);

	vector<vector<int>> generator_matrix;

	for (int i = 0; i < n; i++) {
		vector<int> new_generator_row;

		while (true) {
			new_generator_row.clear();

			for (int j = 0; j < m; j++)

	new_generator_row.push_back(bool_distribution(gen));

			bool found = false;
			for (vector<int> generator_row :
generator_matrix) {
				if (new_generator_row == generator_row) {
					found = true;
					break;
				}
			}

			if (!found)
				break;
		}

		generator_matrix.push_back(new_generator_row);
```

```
                for (int value : new_generator_row)
                        output << value << " ";
                output << "\n";
        }

        output.close();
}

void determine_characteristics(string filename) {
        ifstream input(filename);

        int n, m;

        input >> n >> m;

        vector<vector<int>> generator_matrix;

        for (int i = 0; i < n; i++) {
                vector<int> generator_row;
                for (int j = 0; j < m; j++) {
                        int value;
                        input >> value;
                        generator_row.push_back(value);
                }
                generator_matrix.push_back(generator_row);
        }

        input.close();

        cout << "Filename : " << filename << "\n";
        cout << "Generator matrix (n=" << n << ", m=" << m << ")\n";

        for (vector<int> generator_row : generator_matrix) {
                for (int value : generator_row)
                        cout << value << " ";
                cout << "\n";
        }
        cout << "\n";

        int code_dimension = n;
        cout << "Code dimension = " << code_dimension << "\n";

        //int code_length = m;
        //cout << "Code length = " << code_length << "\n";

        int number_of_code_words = 1 << n;
        cout << "Number of code words = " << number_of_code_words <<
"\n";

        //double block_code_redundancy = (double) m / n;
```

```
        //cout << "Block code redundancy = " << block_code_redundancy
<< "\n";

        //double code_speed = (double) n / m;
        //cout << "Code speed = " << code_speed << "\n";

        int minimum_code_distance = m;

        for (int i = 0; i < n; i++) {
                for (int j = i + 1; j < n; j++) {
                        int code_distance = 0;

                        for (int k = 0; k < m; k++)
                                if (generator_matrix[i][k] !=
generator_matrix[j][k])
                                        code_distance++;

                        if (code_distance < minimum_code_distance)
                                minimum_code_distance = code_distance;
                }
        }

        cout << "Minimum code distance = " << minimum_code_distance <<
"\n";

        cout << "\n\n";
}

int main() {
        for (int i = 1; i <= 5; i++) {
                string filename = "matrix_file_" + to_string(i) +
".txt";
                generate_file(filename);
                determine_characteristics(filename);
        }

        return 0;
}
```