Федеральное государственное автономное

образовательное учреждение

высшего профессионального образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9**

**«Элементарное кодирование повторов»**

Преподаватель 27.05.2016 Пушкарёв К.В.

подпись, дата инициалы, фамилия

Студент КИ15-08Б 27.05.2016 Войченко В.В.

подпись, дата инициалы, фамилия

Красноярск 2016

**Цели работы:**

1.Изучение сжатия данных.

2.Получение навыков работы с графическими данными в MATLAB.

**Порядок выполнения работы:**

1.Выполнить все задания.

2.Продемонстрировать выполнение заданий преподавателю.

3.Подготовить отчёт.

4.Защитить лабораторную работу перед преподавателем.

**Указания:**

1. Работу выполнять индивидуально.

2. Данные для анализа взять из электронного курса.

3. Обратите внимание на функции MATLAB для работы с графическими форматами: **imread()** — чтение из файла, **imwrite()** — запись в файл, **image()** или imshow() — вывод на экран. Для этих функций двухцветное изображение — это матрица с элементами типа **logical** (1 — белый цвет, 0 — чёрный).

В MATLAB закрыть все открытые в данный момент с помощью **fopen()** файлы можно командой **fclose all**.

4. Для приведения элементов массива к типу **logical** используется функция **logical()**.

5. Для декодирования может быть полезна функция **sscanf()**.

6. Считать, что 1 пиксел изображения и 1 текстовый символ занимают по 1 байту.

7. Количество символов в ячейковом массиве строк C можно определить выажением **length([C{:}]).**

8. **Внимание!** Перед началом работы генератор случайных чисел MATLAB необходимо инициализировать (см. «Генератор случайных чисел MATLAB и его инициализация»).

9. При сжатии данных использовать кодовые слова вида <N><C>, где <N> — количество повторов, <C> — цвет точки ('W' — белый, 'B' — чёрный). Например, 5W = 11111 (5 белых точек), 3B = 000 (3 чёрные точки).

10. Заготовка для функций **rle\_compress(),** **rle\_decompress()** приложена к заданию.

**Задания**

1. Написать функцию, сжимающую двухцветное изображение с помощью кодирования повторов (run-length encoding, RLE):

**out = rle\_compress(data),**

где data — сжимаемые данные (матрица); out — результат (ячейковый массив строк).

2. Написать функцию, декодирующую двухцветное изображение, сжатое с помощью кодирования повторов (run-length encoding, RLE):

**out = rle\_decompress(data)**,

где data — декодируемые данные (ячейковый массив строк); out — результат (матрица).

3. С помощью **rle\_compress()** сжать изображения, приложенные к заданию. Определить средний коэффициент сжатия. Декодировать сжатые изображения с помощью **rle\_decompress()** и проверить, что декодированное изображение равно первоначальному (до сжатия).

4. Сгенерировать 100 случайных двухцветных изображений 500 × 500 пикселов. Оценить для них среднее сжатие. Сравнить с результатом для неслучайных примеров. Сделать выводы.

**Результаты работы:**

Отчёт, включающий программный код, результаты проверки, результаты выполнения заданий, выводы.

1. Функция **rle\_decompress :**

function out = rle\_decompress(data)

out = [];

for i = 1:length(data)

s = sscanf(data{i}, '%d%c');

row = [];

for j = 1:2:length(s)

row = [row parse\_codeword(s(j:j+1))];

end

out = [out; row];

end

1. Функция **parse\_codeword:**

function r = parse\_codeword(cw)

if cw(2) == 'B'

symbol = logical(0);

else

symbol = logical(1);

end

r = [];

for t = 1:cw(1)

r(t) = symbol;

end

III. Функция **make\_codeword:**

function cw = make\_codeword(rcount, symbol)

if symbol == 1

symbol = 'W';

else

symbol = 'B';

end

cw = [int2str(rcount) symbol];

IV. Функция **rle\_compress:**

function out = rle\_compress(data)

out = {};

for i = 1:size(data, 1) %i строки

prev = data(i, 1);

rcount = 1;

str = '';

for j = 2:size(data, 2) %j столбцы

% Счёт повторов и выдача кодовых слов

if prev == data(i, j)%если в столбце или в строке есть prev, то rcount++

rcount = rcount + 1;

else

str = [str make\_codeword(rcount, prev)];

rcount = 1;

end

prev = data(i, j);

end

if rcount > 0

str = [str make\_codeword(rcount, prev)];

end

out{end + 1} = str;

end

V. Функция **Files\_inner**:

function nothing = Files\_inner (file)

disp ('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

disp ('Работаем с файлом')

disp (file)

data = imread(file);

data1 = rle\_compress(data);

data2 = rle\_decompress(data1);

countAfterCompress = length([data1{:}]);

countBeforeCompress = numel(data);

disp('Коэффициент сжатия файла: ')

disp (countAfterCompress / countBeforeCompress)

if data == data2

disp('Данные восстановлены полностью');

else

disp('Данные восстановлены частично');

end

end

**Код для работы программы:**

%%TASK 3

comp\_files = [];

comp\_files (1)= Files\_inner ('cdio.pbm');

comp\_files (2)=Files\_inner ('img.pbm');

comp\_files (3)=Files\_inner ('rrep0.990.pbm');

comp\_files (4)=Files\_inner ('rrep0.998.pbm');

comp\_files (5)=Files\_inner ('rrep0.999.pbm');

comp\_files (6)=Files\_inner ('sierpinski.pbm');

comp\_files (7)=Files\_inner ('u.pbm');

disp ('-------------------');

disp ('Средняя компрессия по данным на курсах файлах:')

disp (mean (comp\_files))

%% TASK 4

rng ('shuffle');

array =[];

for i = 1:100

data = randi([0 1],500, 500); %геренируем логическую матрицу размерностью 500х500

data1 = rle\_compress(data); %

countAfterCompress = length([data1{:}]);

countBeforeCompress = numel(data);

array(i, 1) = countAfterCompress / countBeforeCompress;

disp(i);

end

disp ('-------------------');

disp ('Средний коэффициент сжатия сгенерированных файлов:')

disp(mean(array)); **Результаты работы программы:**

**Таблица 1**

|  |
| --- |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Работаем с файлом  cdio.pbm  Коэффициент сжатия файла  0.2085  Данные восстановлены полностью  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Работаем с файлом  img.pbm  Коэффициент сжатия файла  0.5778  Данные восстановлены полностью  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Работаем с файлом  rrep0.990.pbm  Коэффициент сжатия файла  0.0238  Данные восстановлены полностью  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Работаем с файлом  rrep0.998.pbm  Коэффициент сжатия файла  0.0113  Данные восстановлены полностью  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Работаем с файлом  rrep0.999.pbm  Коэффициент сжатия файла  0.0097  Данные восстановлены полностью |

**Продолжение таблицы 1**

|  |
| --- |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Работаем с файлом  sierpinski.pbm  Коэфициент сжатия файла  0.1044  Данные восстановлены полностью  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Работаем с файлом  u.pbm  Коэфициент сжатия файла  0.0579  Данные восстановлены полностью  -----------------------------------------------  Средняя компрессия по данным на курсах файлах:  0.1419  -----------------------------------------------  Средний коэффициент сжатия сгенерированных файлов:  1.0031 |

**Таблица 2**

|  |  |
| --- | --- |
| **Коэффициент сжатия сгенерированных файлов** | **Коэффициент сжатия**  **файлов из курса** |
| **1.0031** | **0.1419** |

**Вывод:** в файлах, взятых из курса есть большая вероятность повторения одинакового цвета (т.е. длиннее цепочка идущих подряд либо черных, либо белых цветов), чего нельзя сказать о генерируемых файлах. Из этого следует, что компрессия кодирования повторов (run-length encoding, RLE) неэффективна для генерируемых файлов.