Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

ОТЧЕТ

по лабораторной работе №1

ИССЛЕДОВАНИЕ ТЕХНОЛОГИЙ

ОПТИМАЛЬНОГО КОДИРОВАНИЯ.

Выполнил: ст. гр. ИТб-21

Куркчи А.Э.

Проверила:

Маслова М.А.

Севастополь

2015

# 1. Цель

Изучение особенностей методов статического сжатия данных и алгоритмов простого безизбыточного и оптимального кодирования (Шеннона-Фано, Хаффмана), а также приобретение практических навыков информационных расчетов эффективности работы кодеров/декодеров систем передачи и хранения данных.

2. Постановка задач

Вариант №38

“Не хвались умом, коли берешь все хребтом”

3. Текст программы

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Collections**;**

**import** java**.**util**.**HashMap**;**

**import** java**.**util**.**List**;**

**import** java**.**util**.**Map**;**

public class Main **{**

public static void binaryCode**(**String input**,**int m**,**int V**,**int**[]** freq**)** **{**

int len **=** **(**int**)** Math**.**round**(**Math**.**ceil**(**Math**.**log**(**m**)/**Math**.**log**(**2**)));**

int n **=** 0**;**

Map**<**Character**,**String**>** bits **=** **new** HashMap**<>();**

**for(**int i**=**0**;**i**<**2000**;**i**++)** **{**

**if(**freq**[**i**]** **>** 0**)** **{**

bits**.**put**((**char**)** i**,**dec2bin**(**n**,**len**));**

n**++;**

**}**

**}**

len **\*=** input**.**length**();**

log**(**"\tBinary Code:\n"**);**

log**(**"Data volume: %d\n"**,**len**);**

log**(**"Compression ratio: %.5f\n"**,**len**\***1.0D**/**V**);**

log**(**"Avarage char length: %.5f\n"**,**len**\***1.0D**/**input**.**length**());**

StringBuilder result **=** **new** StringBuilder**();**

**for(**int i**=**0**;**i**<**input**.**length**();**i**++)** **{**

result**.**append**(**bits**.**get**(**input**.**charAt**(**i**)));**

result**.**append**(**" "**);**

**}**

log**(**"%s\n"**,**result**.**toString**());**

**}**

public static void shannonFanoCode**(**String input**,**int m**,**int V**,**int**[]** freq**)** **{**

Map**<**Character**,**String**>** bits **=** **new** HashMap**<>();**

List**<**PairSI**>** list **=** **new** ArrayList**<>();**

**for(**int i**=**0**;**i**<**2000**;**i**++)** **{**

**if(**freq**[**i**]** **>** 0**)** **{**

list**.**add**(new** PairSI**(**String**.**valueOf**((**char**)**i**),**freq**[**i**]));**

**}**

**}**

Collections**.**sort**(**list**);**

Collections**.**reverse**(**list**);**

int len **=** shannonFanoBit**(**bits**,**list**,true);**

log**(**"\tShannon-Fano Code:\n"**);**

log**(**"Data volume: %d\n"**,**len**);**

log**(**"Compression ratio: %.5f\n"**,**len**\***1.0D**/**V**);**

log**(**"Avarage char length: %.5f\n"**,**len**\***1.0D**/**input**.**length**());**

StringBuilder result **=** **new** StringBuilder**();**

**for(**int i**=**0**;**i**<**input**.**length**();**i**++)** **{**

result**.**append**(**bits**.**get**(**input**.**charAt**(**i**)));**

result**.**append**(**" "**);**

**}**

log**(**"%s\n"**,**result**.**toString**());**

**}**

private static int shannonFanoBit**(**final Map**<**Character**,**String**>** bits**,**List**<**PairSI**>** chars**,**boolean up**)** **{**

String bit **=** bits**.**isEmpty**()** **?** "" **:** **(**up **?** "0" **:** "1"**);**

int len **=** 0**;**

**for(**PairSI p **:** chars**)** **{**

len **+=** bit**.**length**()\***p**.**s**;**

String s **=** **(**bits**.**get**(**p**.**f**.**charAt**(**0**))** **==** **null)** **?** "" **:** bits**.**get**(**p**.**f**.**charAt**(**0**));**

bits**.**put**(**p**.**f**.**charAt**(**0**),**s**+**bit**);**

**}**

**if(**chars**.**size**()** **>** 1**)** **{**

double half **=** chars**.**stream**().**map**((**p**)** **->** p**.**s**).**reduce**(**0**,**Integer**::**sum**)/**2.0D**;**

int sum **=** 0**;**

int i **=** 0**;**

**while(**i**<**chars**.**size**()** **&&** Math**.**abs**(**half**-**sum**)** **>** Math**.**abs**(**half**-**sum**-**chars**.**get**(**i**).**s**))** **{**

sum **+=** chars**.**get**(**i**).**s**;**

i**++;**

**}**

len **+=** shannonFanoBit**(**bits**,**chars**.**subList**(**0**,**i**),true);**

len **+=** shannonFanoBit**(**bits**,**chars**.**subList**(**i**,**chars**.**size**()),false);**

**}**

**return** len**;**

**}**

public static void haffmanCode**(**String input**,**int m**,**int V**,**int**[]** freq**)** **{**

Map**<**Character**,**String**>** bits **=** **new** HashMap**<>();**

int len **=** 0**;**

ArrayList**<**PairSI**>** list **=** **new** ArrayList**<>();**

**for(**int i**=**0**;**i**<**2000**;**i**++)** **{**

**if(**freq**[**i**]** **>** 0**)** **{**

list**.**add**(new** PairSI**(**String**.**valueOf**((**char**)**i**),**freq**[**i**]));**

bits**.**put**((**char**)**i**,**""**);**

**}**

**}**

**while(**list**.**size**()** **>** 1**)** **{**

Collections**.**sort**(**list**);**

**for(**char a **:** list**.**get**(**0**).**f**.**toCharArray**())** **{**

bits**.**put**(**a**,**"0"**+**bits**.**get**(**a**));**

len **+=** freq**[**a**];**

**}**

**for(**char b **:** list**.**get**(**1**).**f**.**toCharArray**())** **{**

bits**.**put**(**b**,**"1"**+**bits**.**get**(**b**));**

len **+=** freq**[**b**];**

**}**

list**.**add**(new** PairSI**(**list**.**get**(**0**).**f**+**list**.**get**(**1**).**f**,**list**.**get**(**0**).**s**+**list**.**get**(**1**).**s**));**

list**.**remove**(**0**);**

list**.**remove**(**0**);**

**}**

log**(**"\tHaffman Code:\n"**);**

log**(**"Data volume: %d\n"**,**len**);**

log**(**"Compression ratio: %.5f\n"**,**len**\***1.0D**/**V**);**

log**(**"Avarage char length: %.5f\n"**,**len**\***1.0D**/**input**.**length**());**

StringBuilder result **=** **new** StringBuilder**();**

**for(**int i**=**0**;**i**<**input**.**length**();**i**++)** **{**

result**.**append**(**bits**.**get**(**input**.**charAt**(**i**)));**

result**.**append**(**" "**);**

**}**

log**(**"%s\n"**,**result**.**toString**());**

**}**

public static void main**(**String**[]** args**)** **{**

String input **=** "Не хвались умом, коли берешь все хребтом"**;**

int n **=** input**.**length**();**

int freq**[]** **=** **new** int**[**2000**];**

**for(**int i**=**0**;**i**<**2000**;**i**++)** **{**

freq**[**i**]** **=** 0**;**

**}**

**for(**int i**=**0**;**i**<**input**.**length**();**i**++)** **{**

freq**[**input**.**charAt**(**i**)]++;**

**}**

int m **=** 0**;**

double H **=** 0.0**;**

StringBuilder basicAlphabet **=** **new** StringBuilder**();**

log**(**"\tFrequency table:\n"**);**

**for(**int i**=**0**;**i**<**2000**;**i**++)** **{**

**if(**freq**[**i**]** **>** 0**)** **{**

m**++;**

double p **=** **(**1.0D**\***freq**[**i**])/**n**;**

H **+=** p**\***Math**.**log**(**1.0D**/**p**)/**Math**.**log**(**2**);**

basicAlphabet**.**append**(**"\""**);**

basicAlphabet**.**append**((**char**)**i**);**

basicAlphabet**.**append**(**"\", "**);**

log**(**"\"%s\"(%d):\t%.5f\n"**,(**char**)**i**,**i**,**p**);**

**}**

**}**

log**(**"\n"**);**

int V **=** n**\***8**;**

double Hmax **=** Math**.**log**(**m**)/**Math**.**log**(**2**);**

double I **=** input**.**length**()\***H**;**

double D **=** 1.0 **-** H**/**Hmax**;**

log**(**"%-4s = %5d\n"**,**"n"**,**n**);**

log**(**"%-4s = %5d\n"**,**"V"**,**V**);**

log**(**"%-4s = %11.5f\n"**,**"I"**,**I**);**

log**(**"%-4s = %11.5f\n"**,**"H"**,**H**);**

log**(**"%-4s = %11.5f\n"**,**"Hmax"**,**Hmax**);**

log**(**"%-4s = %11.5f\n"**,**"D"**,**D**);**

log**(**"\n"**);**

log**(**"\tBasic Alphabet:\n"**);**

log**(**"%s: %d\n"**,**"Power"**,**m**);**

log**(**"%s\n"**,**basicAlphabet**.**toString**());**

log**(**"\n"**);**

binaryCode**(**input**,**m**,**V**,**freq**);**

log**(**"\n"**);**

shannonFanoCode**(**input**,**m**,**V**,**freq**);**

log**(**"\n"**);**

haffmanCode**(**input**,**m**,**V**,**freq**);**

log**(**"\n"**);**

**}**

private static void log**(**String format**,**Object**...** args**)** **{**

System**.**out**.**format**(**format**,**args**);**

**}**

private static String dec2bin**(**int n**,**int m**)** **{**

char data**[]** **=** **new** char**[**m**];**

**for(**int i**=**0**;**i**<**m**;**i**++)** **{**

data**[**i**]** **=** '0'**;**

**}**

int i **=** m**-**1**;**

**while(**n **>** 0**)** **{**

**if(**n**%**2 **==** 1**)** **{**

data**[**i**]** **=** '1'**;**

**}**

n **/=** 2**;**

i**--;**

**}**

**return** **new** String**(**data**);**

**}**

private static abstract class Pair**<**A**,**B**>** **implements** Comparable**<**Pair**<**A**,**B**>>** **{**

public final A f**;**

public final B s**;**

public Pair**(**A f**,**B s**)** **{**

**this.**f **=** f**;**

**this.**s **=** s**;**

**}**

**}**

private static class PairSI **extends** Pair**<**String**,**Integer**>** **{**

public PairSI**(**String f**,**Integer s**)** **{**

**super(**f**,**s**);**

**}**

@Override

public int compareTo**(**Pair**<**String**,**Integer**>** t**)** **{**

int a **=** **((**int**)** s**)** **-** **((**int**)** t**.**s**);**

**return** a **==** 0 **?** f**.**compareTo**(**t**.**f**)** **:** a**;**

**}**

**}**

**}**

4. Тестовые примеры

На рисунке 1 представлен вывод программы

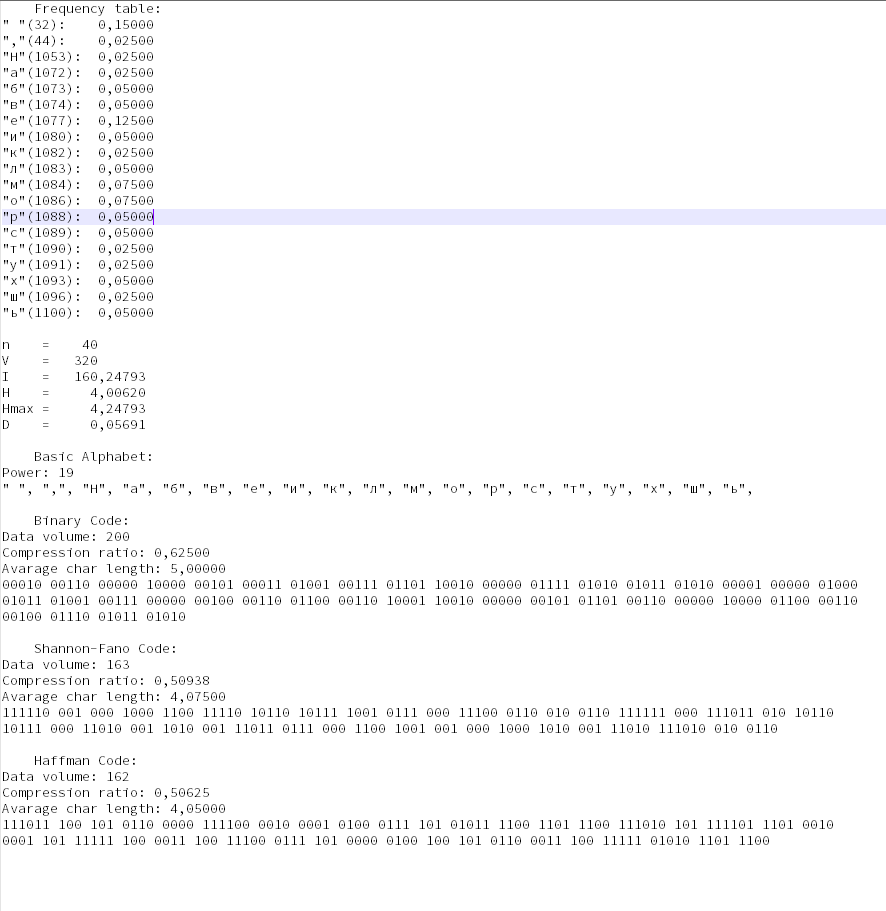


Рисунок 1 – Вывод программы

В разделе «Frequency table» выводится для каждого символа начального алфавита сам этот символ, его код и частота встречаемости. Расчётные значения длины сообщения, объема, количества информации, энтропии сообщения, максимальной энтропии и избыточность выведены в строках под именами n, V, I, H, Hmax и D соответственно. Базовый алфавит расписан в строку в разделе «Basic Alphabet». Для алгоритмов бинарного кодирования, Шенона-Фано и Хаффмана в соответствующих разделах выведены объём итогового сообщения, коэффициент сжатия, средняя длина символа и само закодированное сообщение.

Вывод

В ходе лабораторной работы были изучены особенности методов статического сжатия данных и алгоритмов простого безизбыточного и оптимального кодирования, а также приобрёл практические навыки информационных расчетов эффективности работы кодеров/декодеров систем передачи и хранения данных.