Лабораторная работа № 8. Алгоритмы статистического сжатия информации

**Цель работы:** изучить основные алгоритмы статистическогоо сжатия информации и приобрести практические навыки работы с ними.

1. Краткие теоретические сведения

### 1.1. Алгоритм Шеннона-Фано

**Алгоритм Шеннона — Фано** — один из первых алгоритмов сжатия, который впервые сформулировали американские ученые Клод Шеннон и Роберт Фано. Алгоритм использует коды переменной длины: часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся — кодом большей длины. Коды Шеннона — Фано префиксные, то есть никакое кодовое слово не является префиксом любого другого. Это свойство позволяет однозначно декодировать любую последовательность кодовых слов.

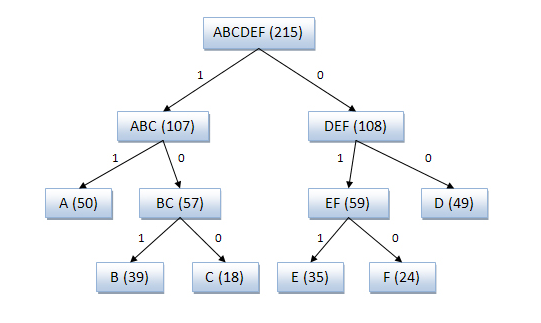
Код Шеннона — Фано строится с помощью дерева. Построение этого дерева начинается от корня. Все множество кодируемых элементов соответствует корню дерева (вершине первого уровня). Оно разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Эти подмножества соответствуют двум вершинам второго уровня, которые соединяются с корнем. Далее каждое из этих подмножеств разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Им соответствуют вершины третьего уровня. Если подмножество содержит единственный элемент, то ему соответствует лист кодового дерева; такое подмножество разбиению не подлежит. Подобным образом поступаем до тех пор, пока не получим все концевые вершины. Ветви кодового дерева размечаем символами 1 и 0.

При построении кода Шеннона — Фано разбиение множества элементов может быть произведено, вообще говоря, несколькими способами. Выбор разбиения на уровне n может ухудшить варианты разбиения на следующем уровне (n + 1) и привести к неоптимальности кода в целом. Другими словами, оптимальное поведение на каждом шаге пути еще не гарантирует оптимальности всей совокупности действий. Поэтому код Шеннона — Фано не является оптимальным в общем смысле, хотя и дает оптимальные результаты при некоторых распределениях вероятностей. Для одного и того же распределения вероятностей можно построить, вообще говоря, несколько кодов Шеннона — Фано, и все они могут дать различные результаты.

**Пример кодового дерева**

Исходные символы:

* A (частота встречаемости 50)
* B (частота встречаемости 39)
* C (частота встречаемости 18)
* D (частота встречаемости 49)
* E (частота встречаемости 35)
* F (частота встречаемости 24)



Кодовое дерево

Полученный код: A — 11, B — 101, C — 100, D — 00, E — 011, F — 010.

Итак, алгоритм формирования сжатого текста можно представить в следующем виде

1. Определяется алфавит символов исходного (сжимаемого) текста.
2. Для каждого символа исходного текста вычисляется частота его встречаемости в тексте.
3. Строится кодирующее дерево следующим образом
   1. В корень дерева записываются все символы и суммарная частота их встречаемости (количество символов в тексте)
   2. Все символы делятся на две группы таким образом, чтобы суммарные часототы встречаемости символов в обеих группах были примерно одинаковы. Эти группы записываются в два потомка корня.
   3. Для каждого из полученных потомков повторяется процедура 3.2 до тех пор пока в группе не останется единственный символ
4. Определяются коды каждого символа следующим образом: двигаясь по кодовому дереву от корня до листа (символа) при переходе на левое поддерево в код добавляется 1, а при переходе на правое поддерево – добавляется 0.
5. Полученные коды сохраняются в таблице (словаре)
6. В исходном тексте символы (байты) заменяются их кодами в соответствии с созданной таблицей.

Пример. Исходный текст

**расколотый\_колокол\_около\_колокольни\_расколот** (44 символа –байта)

Таблица частот символов

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | р | а | с | к | о | л | т | \_ | ы | ь | н | и | й |
| Частота | 2 | 2 | 2 | 7 | 13 | 7 | 2 | 4 | 1 | 1 | 1 | 1 | 1 |

Таблица выделения групп символов с примерно равными частотами встречаемости

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | Частота | 1 | 2 | 3 | 4 | 5 | 6 |
| о | 13 | 20 | 13 | | | | |
| к | 7 | 7 | | | | |
| л | 7 | 24 | 11 | 7 | | | |
| \_ | 4 | 4 | | | |
| р | 2 | 13 | 6 | 2 | | |
| а | 2 | 4 | 2 | |
| с | 2 | 2 | |
| т | 2 | 7 | 3 | 2 | |
| ы | 1 | 1 | |
| ь | 1 | 4 | 2 | 1 |
| и | 1 | 1 |
| й | 1 | 2 | 1 |
| н | 1 | 1 |

Кодовое дерево будет иметь вид

окл\_растыьийн (44)

ок (20)

л\_растыьийн (24)

о (13)

к (7)

л\_(11)

растыьийн (13)

л (7)

\_ (4)

рас (6)

тыьийн (7)

р (2)

ас (4)

ты (3)

ьийн (4)

а (2)

с (2)

т (2)

ы (1)

ьи (2)

йн (2)

ь (1)

и (1)

й (1)

н (1)

Таблица кодов символов

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сим-вол | о | к | л | \_ | р | а | с | т | ы | ь | и | й | н |
| Код | 11 | 10 | 011 | 010 | 0011 | 00101 | 00100 | 00011 | 00010 | 000011 | 000010 | 000001 | 000000 |

Закодированный текст будет иметь вид

0011 00101 00100 10 11 011 11 00011 00010 000001 010 10 11 011 11 10 11 011 010

11 10 11 011 11 010 10 11 011 11 10 11 011 000011 000000 000010

0011 00101 00100 10 11 011 00011 - 135 бит

Коэффициент сжатия

**К сж=44\*8/135=2,61**

Распаковка текста

Для распаковки текста необходимо читать очередной символ сжатого текста и двигаться по дереву кодировки. Если символ 1 – движение влево, если 0 –вправо до тех пор пока не будет найден символ. Найденный символ записывается в распакованный текст, читается следующий символ сжатого текста и движение по кодирующему дереву вновь начинается с корня.

### Алгоритм Хаффмана

### Алгоритм Хаффмана (David A.Huffman) относится к статистическим алгоритмам сжатия. Алгоритм основа на том, что различные символы имеют разные частоты появления в тексте. Если символы с высокой частотой появления закодировать более короткими кодами, а символы, имеющие низкие частоты появления более длинными кодами, то можно уменьшить объем информации по сравнению со случаем кодировки символов кодами одинаковой длины.

### Но при этом необходимо обеспечить чтобы более короткие коды не являлись префиксами более длинных кодов.

### Формирование кодов, отвечающих указанным требованиям возможно при использовании так называемого дерева Хаффмана.

### Дерево кодирования Хаффмана - это двоичное дерево, у которого каждый узел имеет вес, и при этом вес предка равен суммарному весу его потомков.

### Алгоритм Хаффмана для сжатия информации.

### Определить алфавит символов текста.

### Определить частоту появления каждого символа в тексте, разделив количество появлений символа в тексте на общее количество символов в тексте.

### Построить дерево Хаффмана по следующим правилам

### Представить символы текста в виде узлов дерева (листьев) и каждому узлу присвоить числовой ключ, равный частоте появления символа.

### Создать следующий уровень дерева, объединяя имеющиеся узлы попарно, начиная с узлов с наименьшими ключами, в новые узлы. Каждому объединенному узлу присвоить числовой ключ, равный сумме ключей объединяемых узлов.

### Если в результате получен единственный узел, то дерево построено, иначе переход на пункт 3.2.

### Сформировать коды символов по следующему правилу:

### Двигаясь от корня дерева к узлу символа (листу) записывать 0 при движении влево и 1 при движении вправо. Полученная последовательность нолей и единиц и есть код символа.

### Последовательно записать коды символов.

Пример построения дерева кодирования Хаффмана и определения кодов символов.

Алфавит некоторого текста и частоты появления символов представлены в таблице 1.

Таблица 1.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | *Z*1 | Z2 | Z3 | Z4 | Z5 | Z6 | Z7 | Z8 |
| Частота | 0,22 | 0,20 | 0,16 | 0,16 | 0,10 | 0,10 | 0,04 | 0,02 |

Создание узлов (листьев) с ключами, равными частотам появления символов, показано на рисунке 1.

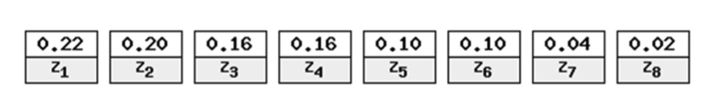


Рис.1. Узлы символов текста с ключами

На первом шаге из листьев дерева выбираются два листа с наименьшими ключами. Такими листьями являются листья символов Z7 (частота равна 0.04) и Z8 (частота равна 0.02). Эти листья объединяются в узел-предок с ключом, равным сумме ключей узлов потомков

**0,04 + 0,02 = 0,06.**  Дерево кодирования после первого шага приведено на рис. 2.

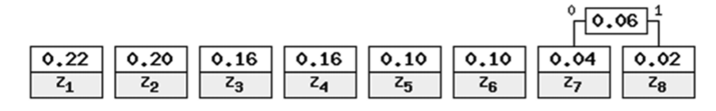


Рис. 2. **Дерево кодирования Хаффмана после первого шага**

На втором шаге наименьшие ключи имеет пара - лист *Z6* (0.1) и свободный узел (Z7 **+** Z8) (0.06). Для них создастся предок с весом **0,16.**  На данном шаге дерево кодирования приведено на рисунке 3.

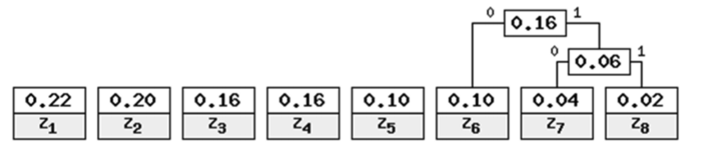


Рис. 3.**Дерево кодирования Хаффмана после второго шага**

На третьем шаге наименьшие ключи имеют узлы Z5 и свободный узел *(* Z6 +Z7 +Z8). На этом шаге можно создать предка для Z5 и *(* Z6 +Z7 +Z8) с ключом **0,26,** получив при этом дерево кодирования, представленное на рис. 4. Обратите внимание, что в данной ситуации возможны несколько вариантов соединения узлов с наименьшими весами. При этом все такие варианты будут правильными, хотя и могут привести к различным наборам кодов.

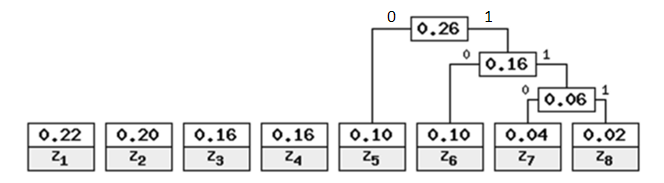


Рис. .4. **Дерево кодирования Хаффмана после третьего шага**

На четвертом парой с наименьшими ключами оказываются листья Z3 и Z4. Дерево кодирования Хаффмана для этого шага приведено на рис. 5.

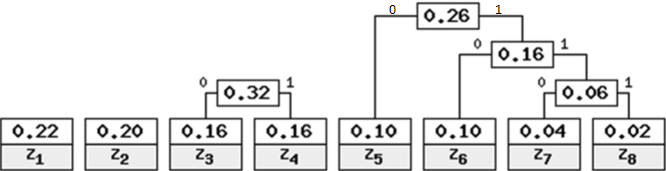


Рис. 5. **Дерево кодирования Хаффмана после четвертого шага**

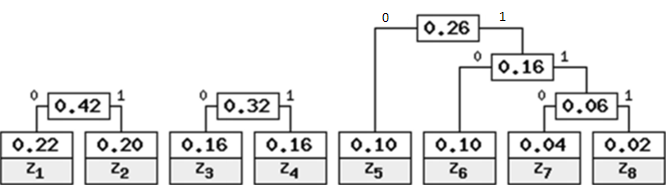


Рис. 6. Дерево кодирования Хаффмана после пятого шага

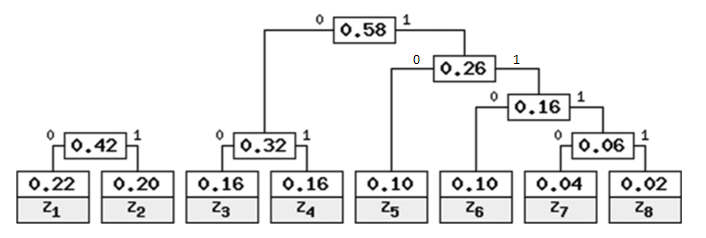


Рис. 7. **Дерево кодирования Хаффмана после шестого шага**

На пятом шаге выбираются узлы с наименьшими ключами Z1 (0,22**)** и Z2 (0,20) и для них создается предок с ключом 0.42**.** Дерево кодирования Хаффмана после пятого шага приведено на рис. 6.

На шестом шаге остается три свободных узла с весами 0,42, 0,32 и 0,26.Выбираются узлы с наименьшими ключами 0,32 и 0,26и для них создается предок с ключом 0.58.Дерево кодирования Хаффмана после шестого шага приведено на рис. 7.

На седьмом шаге остается объединить две оставшиеся свободные вершины, после чего будет получено окончательное дерево кодирования Хаффмана, приведенное на рис. 8. На этом дереве помечено движение влево 0 и движение вправо 1.

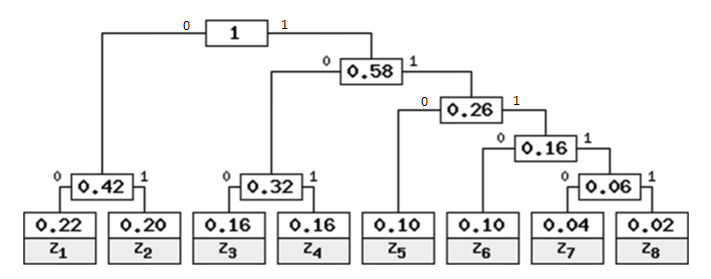


Рис. 8*.* **Окончательное дерево кодирования Хаффмана**

На основании построенного дерева буквы представляются кодами, отражающими путь от корневого узла до листа, соответствующего нужной букве. При движении от корня к листу и записывая соответствующие цифры на каждом шаге можно получить коды Хаффмана для каждого символа. Полученные коды представлены в таблице 2.

Таблица 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Символ | *Z*1 | Z2 | Z3 | Z4 | Z5 | Z6 | Z7 | Z8 |
| Код | 00 | 01 | 100 | 101 | 110 | 1110 | 11110 | 11111 |

Из таблицы 2 видно, что наиболее часто встречающиеся символы закодированы самыми короткими кодами, а наиболее редко встречающиеся - кодами большей длины, причем коды построены таким образом, что ни одна кодовая комбинация не совпадает с началом более длинной комбинации. Это позволяет однозначно декодировать сообщения без использования разделительных символов.

Предположим, что имеющийся текст содержит 1000 символов из рассмотренного алфавита Z1-Z8

При ASCII - кодировании этого текста его размер составит 8000 бит. В соответствии с частотами появления символов в тексте, приведенными в таблице 16.1 и размерами их кодов, приведенными в таблице 16.2 текст из 1000 символов, закодированный кодами Хаффмана будет иметь размер

V=220 \* 2 +200 \* 2 + 160 \* 3 +160 \* 3 +100 \* 3 + 100 \* 4 + 40 \* 5 + 20 \* 5 = 2320 бит

Коэффициент сжатия будет

Ксж=8000/2320 =3.45

**Распаковка текста, сжатого алгоритмом Хаффмана**

Сжатый текст представляет собой последовательность нолей и единиц без какого либо деления на символы. Поэтому для распаковки нужно иметь дерево Хаффмана, по которому этот текст был сжат. Алгоритм распаковки может быть описан следующим образом.

1. Выбирается первый символ сжатого текста.
2. В дереве Хаффмана начинается перемещение от корня к листу (символу).
3. Если символ равен единице то переход по дереву вправо, а если 0, то влево.
4. Если очередной узел дерева не является листом, то выбирается следующий символ сжатого текста и переход к пункту 3.
5. Если очередной узел дерева является листом, то считывается символ из этого листа и помещается в распакованный текст.
6. Если в сжатом тексте есть символы, то выбирается следующий символ сжатого текста и переход к пункту 2.
7. Если в сжатом тексте больше нет символов, то КОНЕЦ.

Поскольку само дерево Хаффмана имеет небольшие размеры, то его сохранение вместе со сжатым текстом несущественно уменьшает коэффициент сжатия.

## **2. Арифметическое кодирование**

Основная идея заключается в том, чтобы присваивать коды не отдельным символам, а их последовательностям.  
1. Определяются частоты появления символов в сообщении.

2. Выбирается рабочий отрезок [0; 1)

3. На рабочем отрезке откладываются точки так, что полученные длины отрезков пропорциональны частоте использования символов.

4. Если есть очередной символ в сообщении, то он выбирается и определяется соответствующий ему отрезок, который считается рабочим отрезком. Переход к п.3

5. Если символов больше нет, то выбирается любое число в пределах полученного отрезка, которое является кодом сообщения.

Пример.

Кодирование текстовой строки **«*МАТЕМАТИКА***».

Алфавит кодируемого сообщения содержит следующие символы:

{***М***, ***А***, ***Т***, ***Е***, ***И***, ***К***}.

Частоты символов и соответствующие им отрезки приведены в Таблице 3.

Символы в таблице символов и интервалов можно располагать в любом порядке: по мере их появления в тексте, в алфавитном или по возрастанию вероятностей – это не принципиально. Результат кодирования может быть разным, но эффект будет одинаковым.

***Таблица 3.***

|  |  |  |
| --- | --- | --- |
| **Символ** | **Вероятность** | **Интервал** |
| ***М*** | ***0,2*** | **[0; 0,2)** |
| ***А*** | ***0,3*** | **[0,2; 0,5)** |
| ***Т*** | ***0,2*** | **[0,5; 0,7)** |
| ***Е*** | ***0,1*** | **[0,7; 0,8)** |
| ***И*** | ***0,1*** | **[0,8; 0,9)** |
| ***К*** | ***0,1*** | **[0,9; 1,0)** |

После просмотра первого символа сообщения «***М***» кодер сужает исходный интервал **[0; 1)** до нового **[0; 0,2)**. Таким образом, после кодирования первой буквы результат кодирования будет находиться в интервале **[0; 0,2)**.

Следующий символ «***А***» кодируется ***подынтервалом*** ***внутри интервала, выделенного для предыдущих символов***, сужая его до **[0,04; 0,1)** - *верхняя и нижняя границы нового интервала определяются как начало предыдущего интервала плюс произведение ширины этого интервала на значения границ отрезка, отвечающего текущему символу* (*табл. 1*): ***lowi***=0+0,2×0,2=**0,04**; ***high***=0+0,5×0,2=**0,1**).

Следующий символ, поступающий на вход кодера, – это буква «***Т***» (символу «***Т***» соответствует отрезок **[0,5; 0,7)** в таблице кодера (*табл. 1*)). Применительно к уже имеющемуся рабочему интервалу сообщения, состоящего из предыдущих букв, новый интервал будет **[0,07; 0,082)** (***low***=0,04+0,6×0,5=**0,07**; ***high***=0,04+0,6×0,7=**0,082**) и т. д.

Таким образом, последовательность интервалов, соответствующих кодируемому сообщению, будет следующей (*табл. 4*).

***Таблица 4.***

| **Символ – интервал** | **Интервал сообщения** | **Ширина интервала** |
| --- | --- | --- |
| ***М* - [0; 0,2)** | **[0; 0,2)** | ***0,2*** |
| ***А* - [0,2; 0,5)** | **[0,04; 0,1)** | ***0,6*** |
| ***Т* - [0,5; 0,7)** | **[0,07; 0,082)** | ***0,012*** |
| ***Е* - [0,7; 0,8)** | **[0,0784; 0,0796)** | ***0,0012*** |
| ***M* - [0; 0,2)** | **[0,0784; 0,07864)** | ***0,00024*** |
| ***А* - [0,2; 0,5)** | **[0,078448; 0,07852)** | ***0,72´10-4*** |
| ***Т* -****[0,5; 0,7)** | **[0,078484; 0,0784984)** | ***0,144´10-4*** |
| ***И*** -**[0,8; 0,9)** | **[0,07849552; 0,07849696)** | ***0,144´10-5*** |
| ***К* -****[0,9; 1,0)** | **[0,078496816; 0,07849696)** | ***0,144´10-6*** |
| ***А* -**[0,2; 0,5) | **[0,0784968448; 0,078496888)** | ***0,432´10-7*** |

Результат кодирования сообщения «***МАТЕМАТИКА***» - это вещественное число, принадлежащее интервалу **[0,0784968448; 0,078496888)**. Целое число, деленное на минимальную степень 2, принадлежащее данному отрезку, - это

**0,07849687=1316959/224**

Двоичный 24-разрядный код числа **1316259**10=0001010000011000010111112. Этот код и будет арифметическим кодом сообщения

***Code*(*МАТЕМАТИКА*)**=000101000001100001011111.

**К сж=10/3=3.3**

**Декодирование**

**Декодеру**, как и кодеру, известна таблица распределения отрезков, выделенных символам алфавита источника. Декодирование арифметического кода сообщения происходит по следующему алгоритму:

**1.** По таблице отрезков символов алфавита определяется интервал, содержащий текущий код сообщения – и по этому интервалу из той же таблицы однозначно определяется символ исходного сообщения. Если это маркер конца сообщения, то конец, иначе – переход к шагу 2.

**2.**  Из текущего кода вычитается нижняя граница содержащего его интервала. Полученная разность делится на длину этого интервала. Полученное значение считается новым текущим кодом. Переход к шагу 1.

Пример.

Длина исходного сообщения 10 символов, длина кода 24 бита. Двоичный арифметический кодсообщения 0001010000011000010111112=**1316959**10.

Вещественное число, принадлежащее интервалу, однозначно определяющему закодированное сообщение,

**1316959/224 =0,078449687**

Это число и будет ***текущим кодом*** сообщения.

По исходной таблице значений частот и назначенных им интервалов (*таблица* 1) определяется отрезок, которому принадлежит это число, - **[0; 0,2)**, и соответственно, что первый закодированный символ – это «***М***».

Исключим из результата кодирования влияние теперь уже известного первого символа «***М***»: *для этого вычтем из текущего кода нижнюю границу отрезка раскодированного символа сообщения и разделим полученный результат на ширину этого отрезка*, т.е. следующим декодируемым числом будет .

http://bookwu.net/imgs/1412338656image098.gif

Это число принадлежит отрезку **[0,2; 0,5)**, отведенному символу «***А***», следовательно, вторым символом декодированной последовательности будет «***А***».

Исключим из интервала **[0,2; 0,5)** влияние буквы «***А***». Для этого вычтем из текущего кода нижнюю границу этого интервала и разделим на его ширину:

http://bookwu.net/imgs/1412338656image099.gif

Результат принадлежит отрезку **[0,5; 0,7)** символа «***Т***» - это очередной декодируемый символ.

Исключив из результата декодирования влияние буквы «***Т***», получим

http://bookwu.net/imgs/1412338656image100.gif

Результат принадлежит отрезку буквы «***Е***» **[0,7; 0,8)** и т. д., пока не декодируем все символы (*табл. 5*).

***Таблица 5***

|  |  |  |  |
| --- | --- | --- | --- |
| **Декодируемое число** | **Символ на выходе** | **Интервал** | **Ширина интервала** |
| **0,07849687** | ***М*** | **[0; 0,2)** | *0,2* |
| **0,39248435** | ***А*** | **[0,2; 0,5)** | *0,3* |
| **0,6416145** | ***Т*** | **[0,5; 0,7)** | *0,2* |
| **0,7080725** | ***Е*** | **[0,7; 0,8)** | *0,1* |
| **0,080725** | ***М*** | **[0; 0,2)** | *0,2* |
| **0,403625** | ***А*** | **[0,2; 0,5)** | *0,3* |
| **0,67875** | ***Т*** | **[0,5; 0,7)** | *0,2* |
| **0,89375** | ***И*** | **[0,8; 0,9)** | *0,1* |
| **0.9375** | ***К*** | **[0,9; 1,0)** | *0,1* |
| **0.375** | ***А*** | **[0,2; 0,5)** | *0,3* |

1. Порядок выполнения работы
2. Создать текстовый файл и записать в него свою фамилию, имя и отчество.
3. В соответствии с алгоритмами Шеннона-Фано и Хафмана закодировать текст.

(Создать деревья Шеннона-Фано и Хафмана)

1. Оценить коэффициент сжатия текста для каждого алгоритма.
2. Закодировать с помощью арифметического алгоритма свою фамилию.

Содержание отчета о лабораторной работе

1. Титульный лист
2. Цель работы и задание для выполнения
3. Содержание текстового файла (Фамилия, имя и отчество).
4. Таблицы и деревья для алгоритмов Шеннона-Фано и Хафмана)
5. Текст, закодированный по каждому алгоритму и оценка коэффициента сжатия информации.
6. Таблицы сжатия и распаковки текста по арифметическому алгоритму
7. Выводы по лабораторной работе.

Контрольные вопросы

1.На чем основаны алгоритмы сжатия информации?

2. Как определяется коэффициент сжатия информации?

3. Как производится сжатие и распаковка текста в алгоритме Шеннона-Фано?

4. Как производится сжатие и распаковка текста в алгоритме Хафмана?

5. Как производится сжатие и распаковка текста в арифметическом алгоритме?