Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчёт

по лабораторной работе №2

«Исследование динамических методов сжатия данных: Хаффмана,

Лемпеля-Зива (LZ), Лемпеля-Зива-Велча ( LZW), RLE»

Выполнил:

ст. гр. ИС-21

Куркчи А.Э.

Проверила:

Маслова М.А

Севастополь

2015

1. Цель работы

Исследование статических методов сжатия данных: Шеннона-Фано, Хаффмана, арифметического.

1. Задание к лабораторной работе

1. Составить текстовое сообщение на естественном языке, содержащее 15-25

символов, среди которых должны быть повторяющиеся.

2. Составить таблицу первичного алфавита источника на основе сообщения п.1 (модель Бернулли ) с указанием частоты встречаемости символов.

3. Рассчитать информационные параметры источника и сообщения: количество информации, энтропию, избыточность.

4. Закодировать сообщение п.1 двоичным безизбыточным кодом.

5. Закодировать сообщение п.1 динамическим кодом Хаффмана.

6. Закодировать сообщение п.1 кодом LZ.

7. Закодировать сообщение п.1 кодом LZW.

8. Рассчитать среднюю длину символа сообщения и определить коэффициенты сжатия для методов п.п.4-7.

9. Декодировать сообщения п.п. 4-7.

10. Составить пример черно-белого бинарного изображения размером 8х8 пикселей и закодировать его по методу RLE.

11. Сделать выводы по работе.

1. Ход работы
2. Сообщение, которое следует зашифровать: Не\_хвались\_умом,\_коли.

Количество символов = n = 22.

Количество неповторяющихся символов = m = 15.

1. Таблица первичного алфавита с указанием вероятности встречаемости символов. Рассчитаем информационные параметры источника и сообщения: количество информации, энтропию, избыточность.

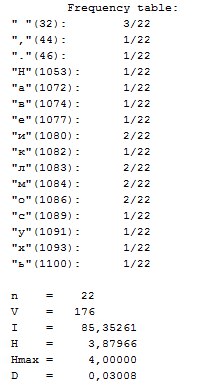


Рисунок 1 – Расчет частоты встречаемости символа и информационных параметров.

1. Кодирование сообщения двоичным безизбыточным кодом.

Для кодирования сообщения данным способом каждому символу первичного алфавита необходимо присвоить некое число, выраженное в виде двоичного кода.

|  |  |
| --- | --- |
| Литера | Код символа |
| Н | 0000 |
| е | 0001 |
| \_ | 0010 |
| х | 0011 |
| в | 0100 |
| а | 0101 |
| л | 0110 |
| и | 0111 |
| с | 1000 |
| ь | 1001 |
| у | 1010 |
| м | 1011 |
| о | 1100 |
| , | 1101 |
| к | 1110 |
| . | 1111 |

Таблица 1 – Кодирование литер.

Исходное сообщение: Хлеб да вода - крестьянская еда.

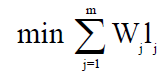
Закодированное сообщение: 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 0010 1010 1011 1100 1011 1101 0010 1110 1100 0110 0111 1111

Количество символов = 22; длина сообщения = 88 бит – 44/44 – («0»/«1»).

Средняя длина символа в сообщении = 88/22 = 4 бит/символ.

Коэффициент сжатия = длина закодированного сообщения / длину сообщения, зашифрованного двоичным безизбыточным кодом.

1. Кодирование сообщения динамическим кодом Хаффмана.

Суть алгоритма – построение дерева Хаффмана с минимальным весом пути:

(1)

Добавляем в дерево Хаффмана начальный лист с escape\_символом. Символ “Н” с ASCII-кодом 205 не найден в дереве. Новый символ “Н” кодируется кодословом escape. Добавляем Новый символ в дерево. Увеличиваем вес вершины 2 и переходим к ее родителю. Дерево не требует обновлений. Увеличиваем вес вершины 1(далее переходим к родителю, если она не является корнем дерева, как в данном случае).

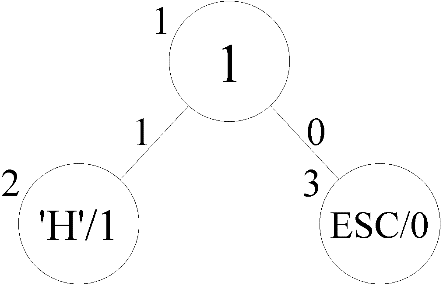


Рисунок 2 – Первый этап построения.

Далее аналогично добавляем последовательно новые листы пока не встретим символ, после добавления которого, необходимо обновление дерева.

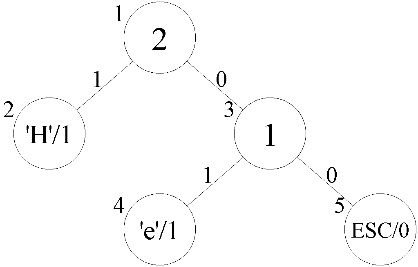
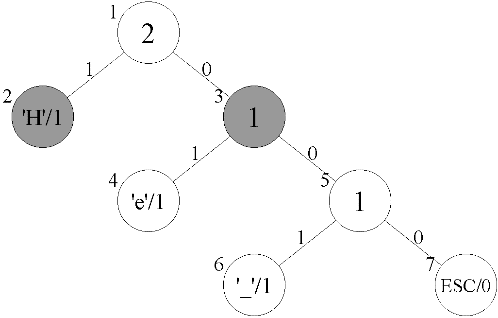
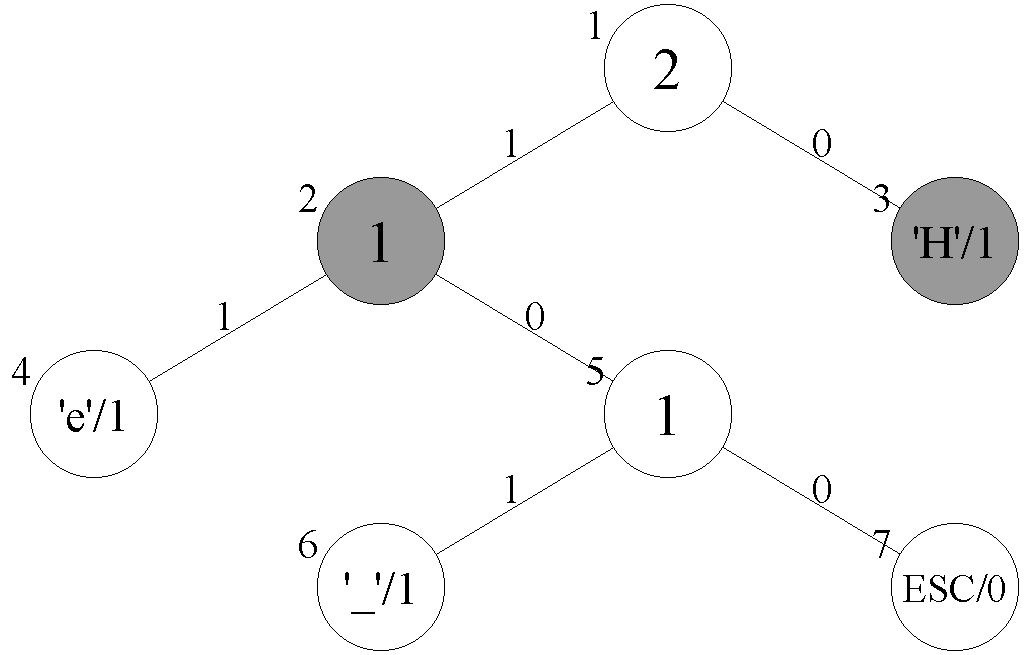


Рисунок 3 – Второй этап построения.

После добавления символа “\_” в дерево дерево не требует обновлений. Увеличиваем вес вершины 6 и переходим к ее родителю. После увеличения веса вершины 5 и перехода к ее родителю находим минимальную вершину 2, равную по весу исходной вершине 3 и не являющуюся ее отцом. Обмениваем поддеревья вершин 2 и 3. Увеличиваем вес вершины 2 и переходим к ее родителю, дерево не требует обновления. Увеличиваем вес вершины 1.



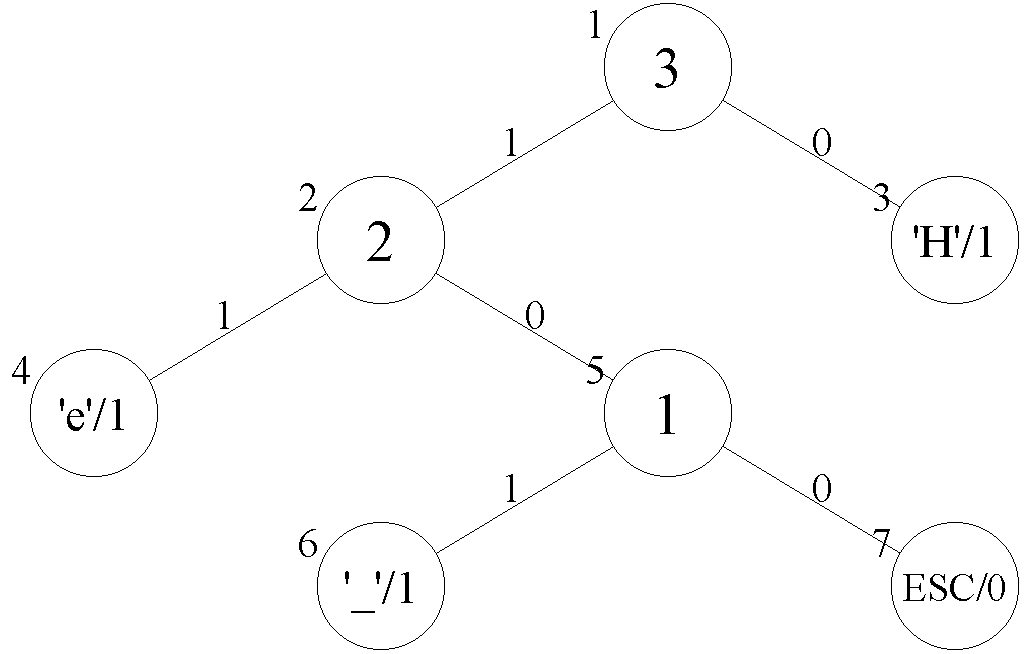


Рисунок 4 – Третий этап построения.

Далее рассуждаем аналогично.

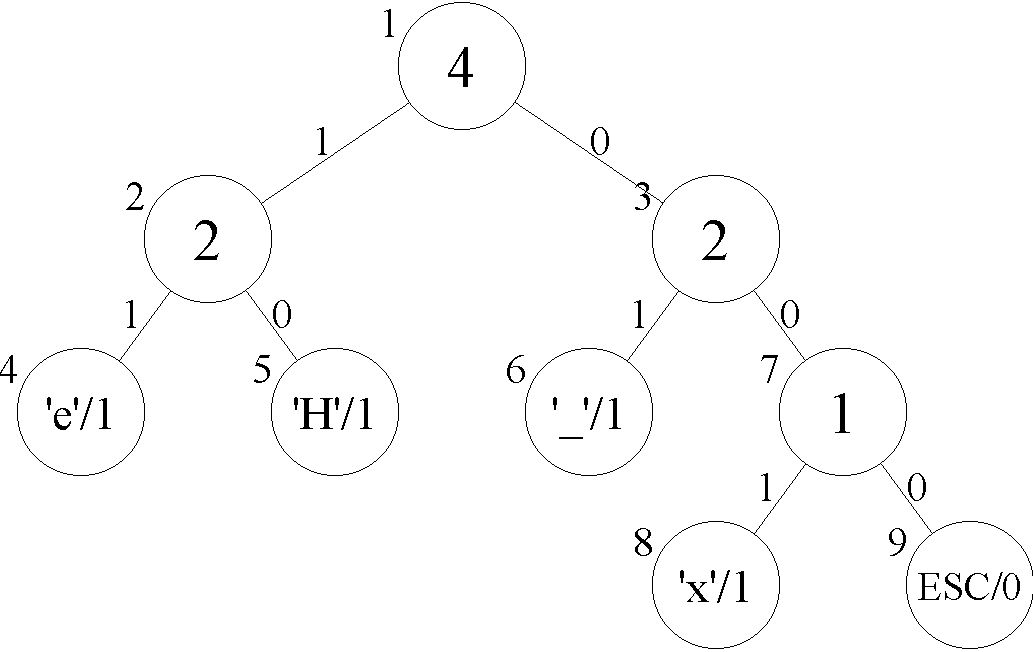


Рисунок 5 – Четвертый этап построения.

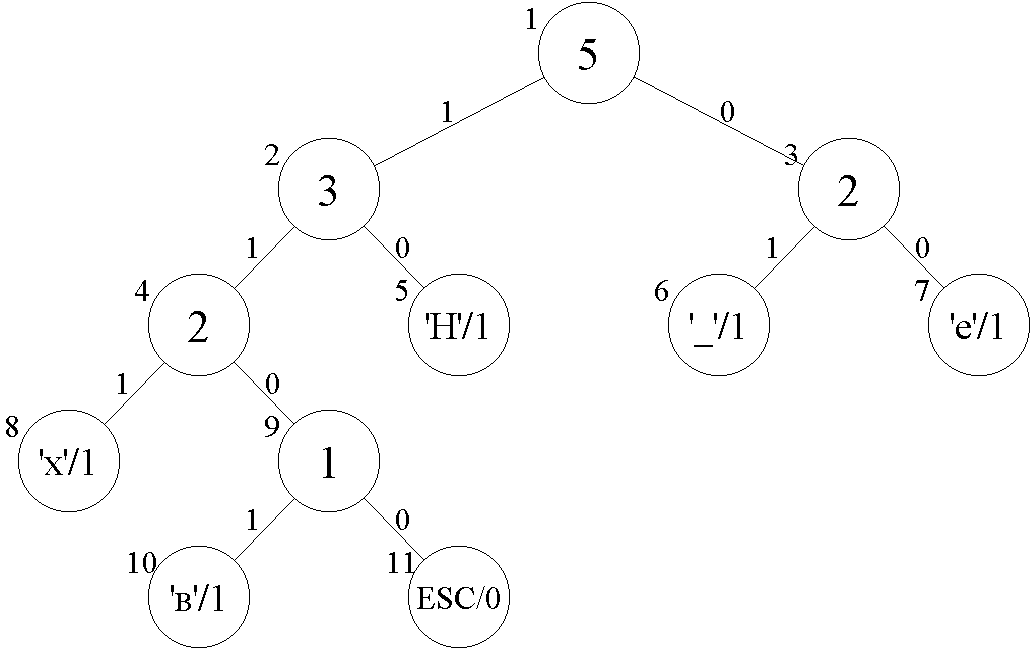


Рисунок 6 – Пятый этап построения.

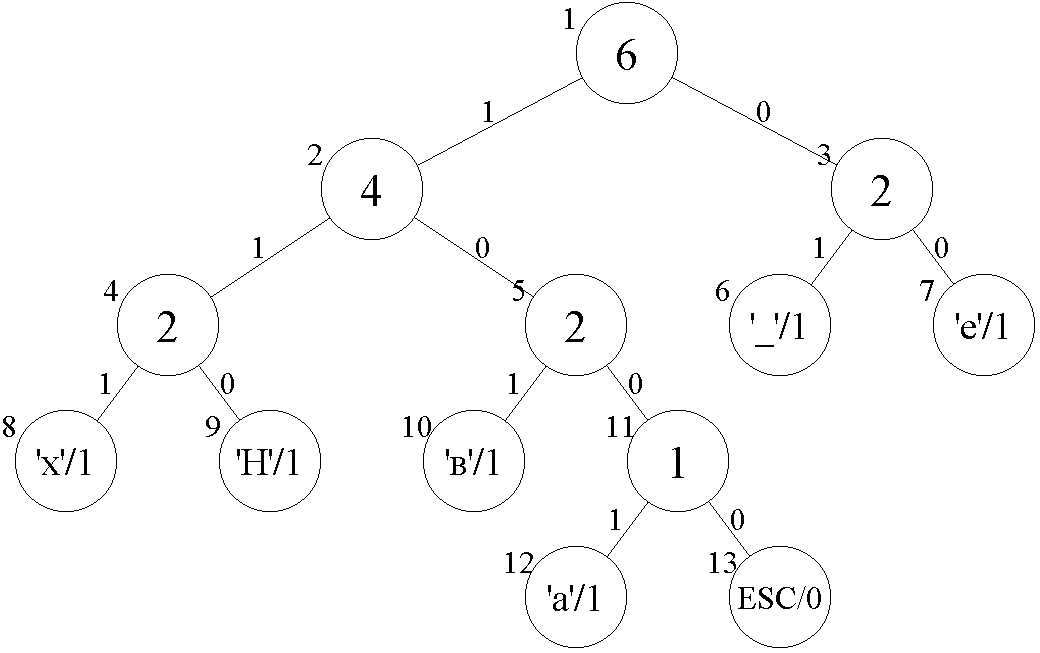


Рисунок 7 – Шестой этап построения.

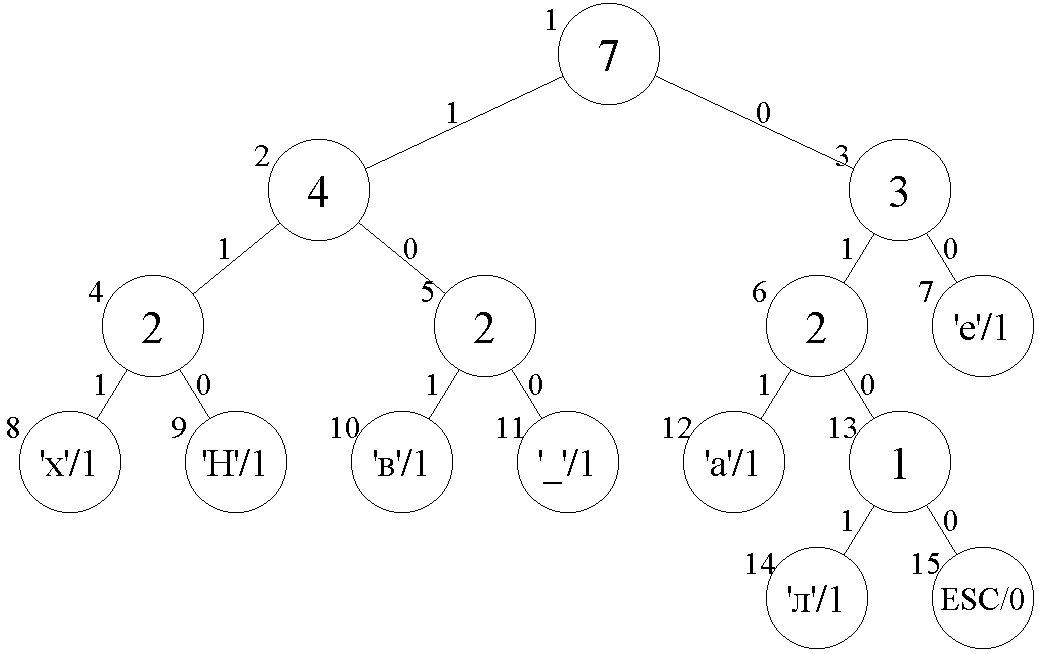


Рисунок 8 – Седьмой этап построения.

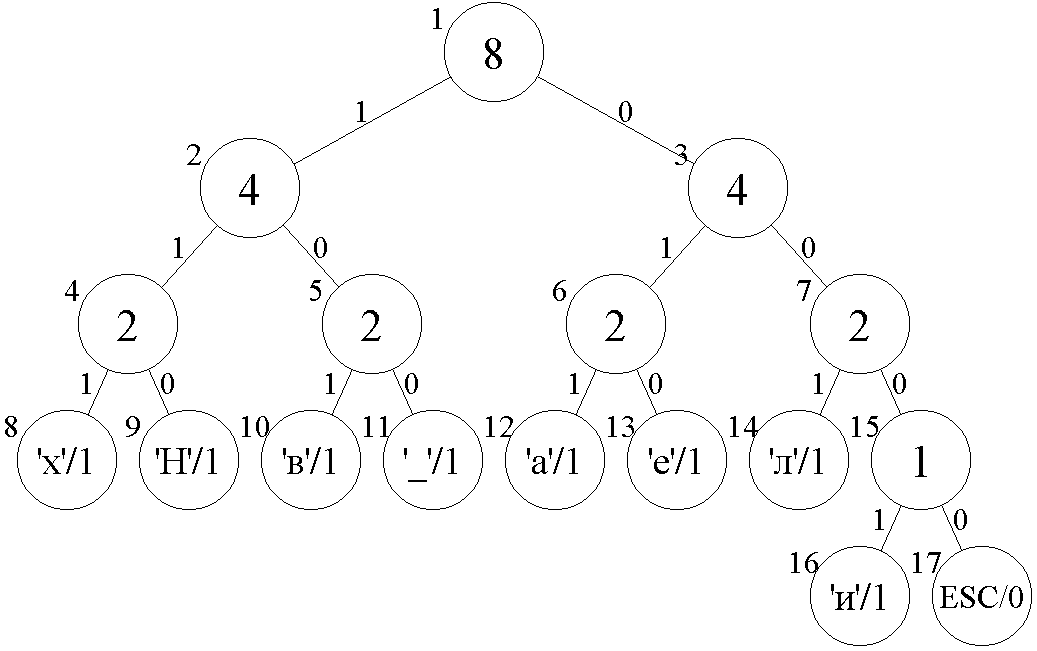


Рисунок 9 – Восьмой этап построения.

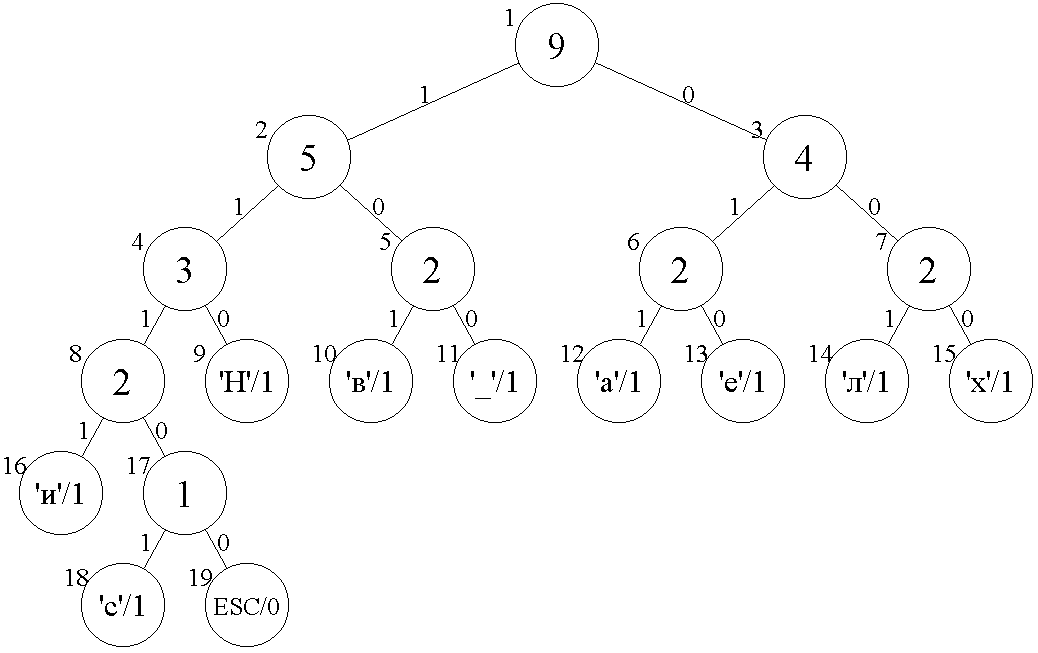


Рисунок 10 – Девятый этап построения.

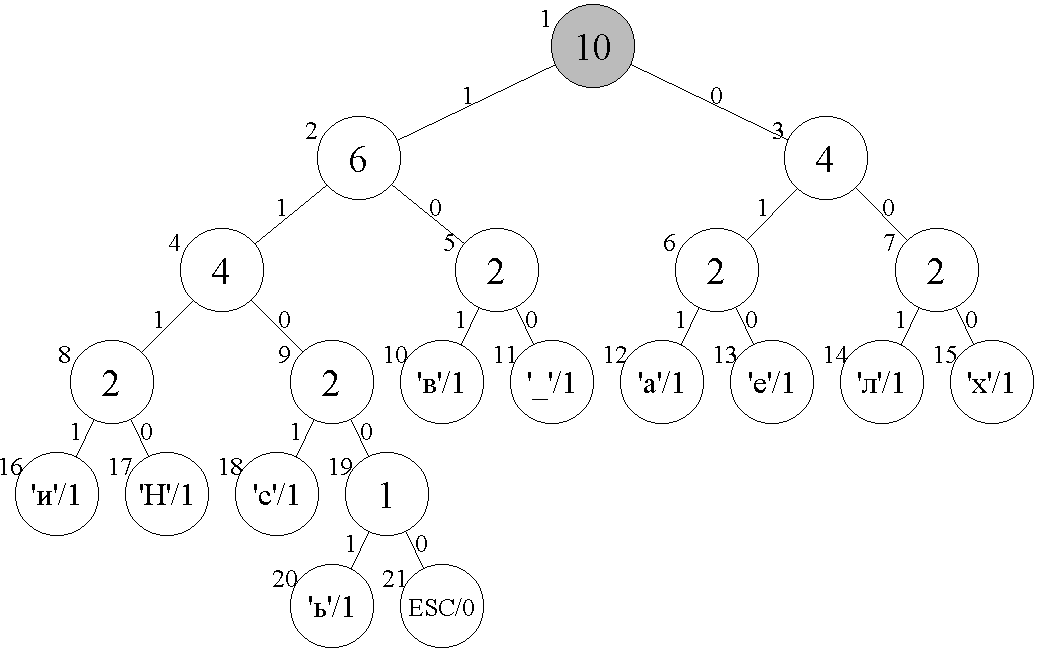


Рисунок 11 – Десятый этап построения.

Символ “\_” с ASCII-кодом 95 найден в дереве. Символ “\_” кодируется своим двоичным представлением в дереве Хаффмана. Нашли минимальную вершину 10, равную по весу исходной вершине 11 и не являющуюся ее родителем. Обмениваем поддеревья вершин 10 и 11. Увеличиваем вес вершины 10 и переходим к ее родителю. Дерево не требует обновления, увеличиваем вес вершины 2 и переходим к ее родителю. Увеличиваем вес вершины-корня.

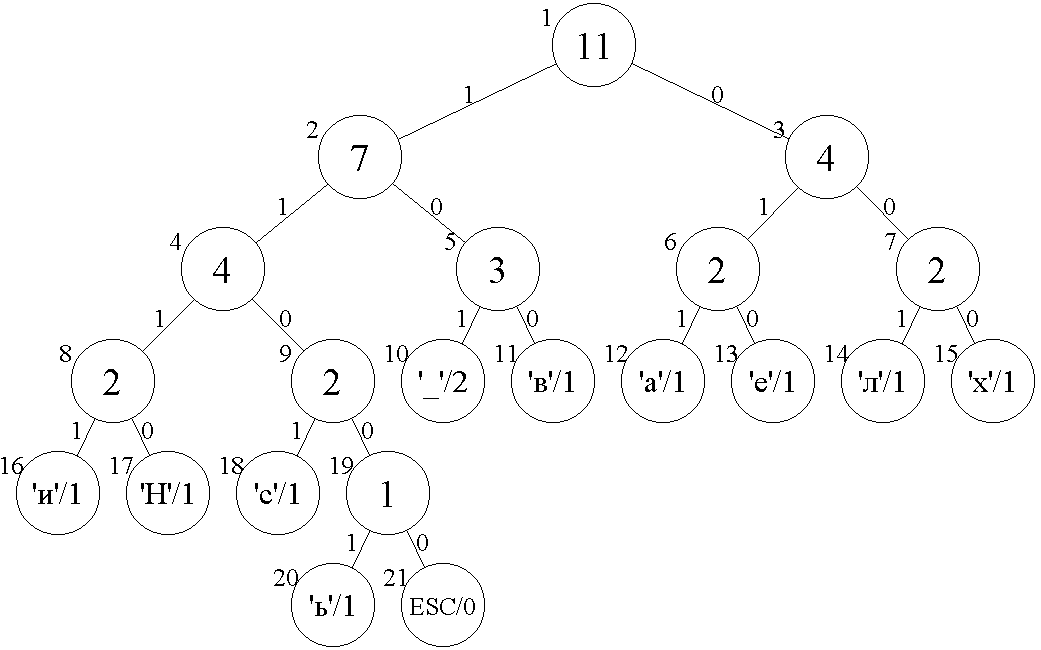


Рисунок 12 – Одиннадцатый этап построения.

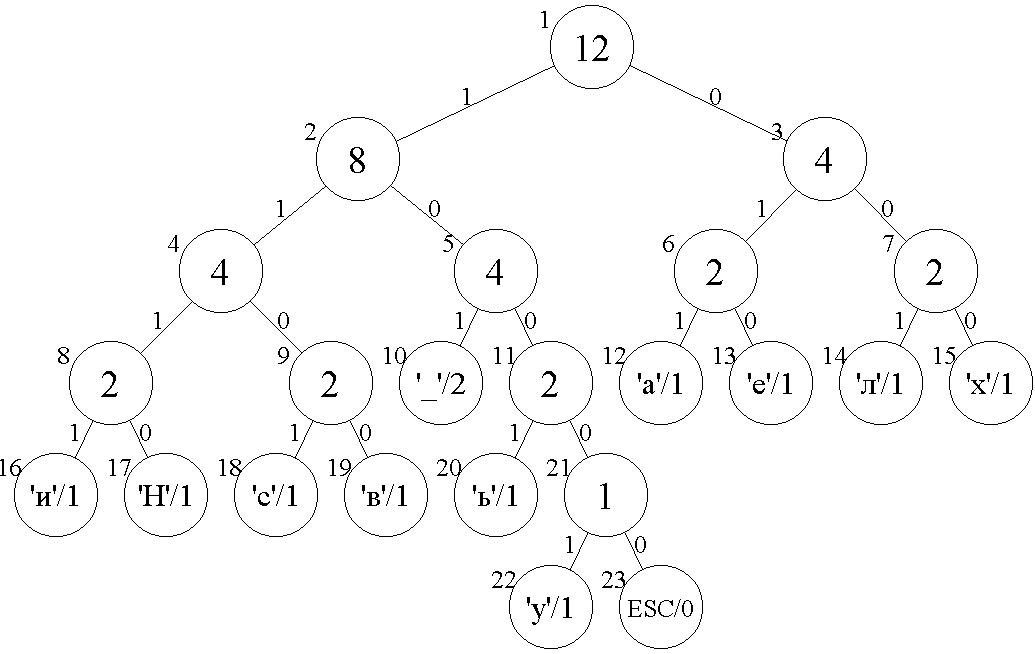


Рисунок 13 – Двенадцатый этап построения.

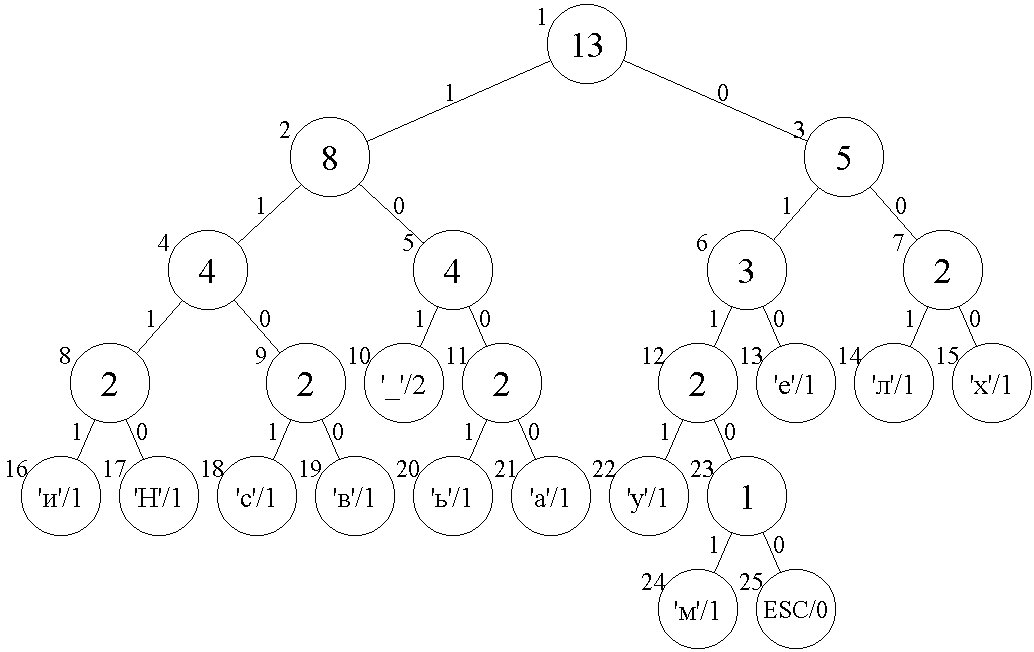


Рисунок 14 – Тринадцатый этап построения.

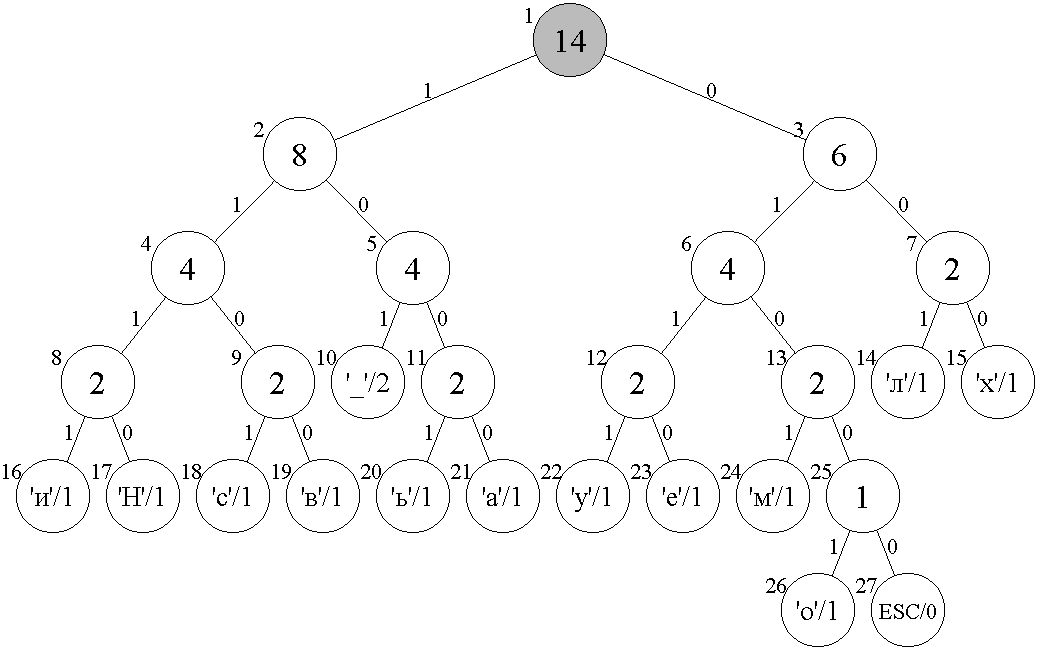


Рисунок 15 – Четырнадцатый этап построения.

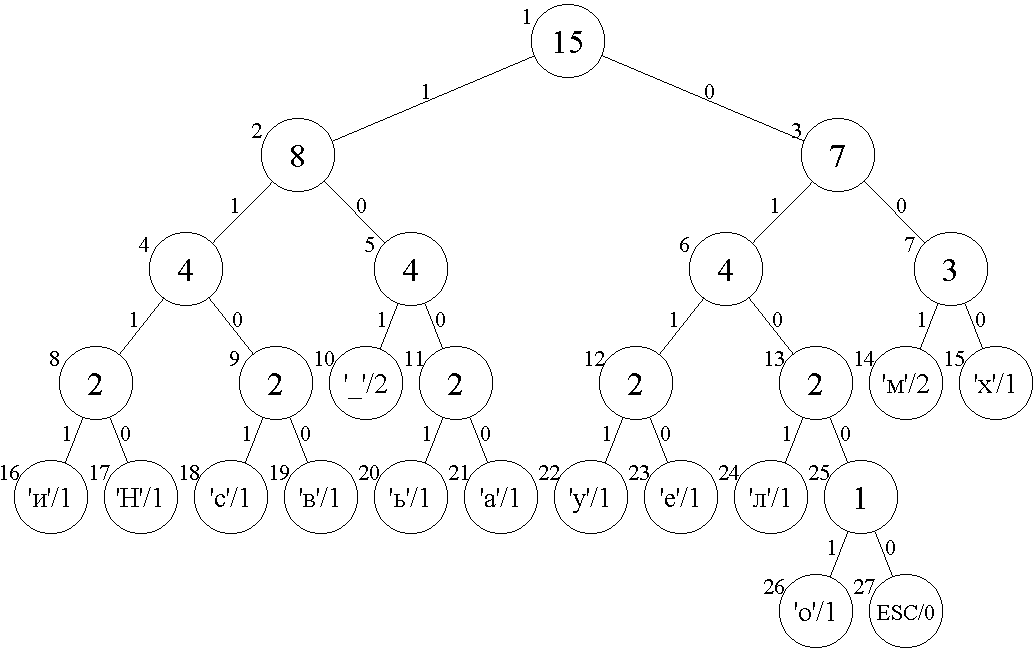


Рисунок 16 – Пятнадцатый этап построения.

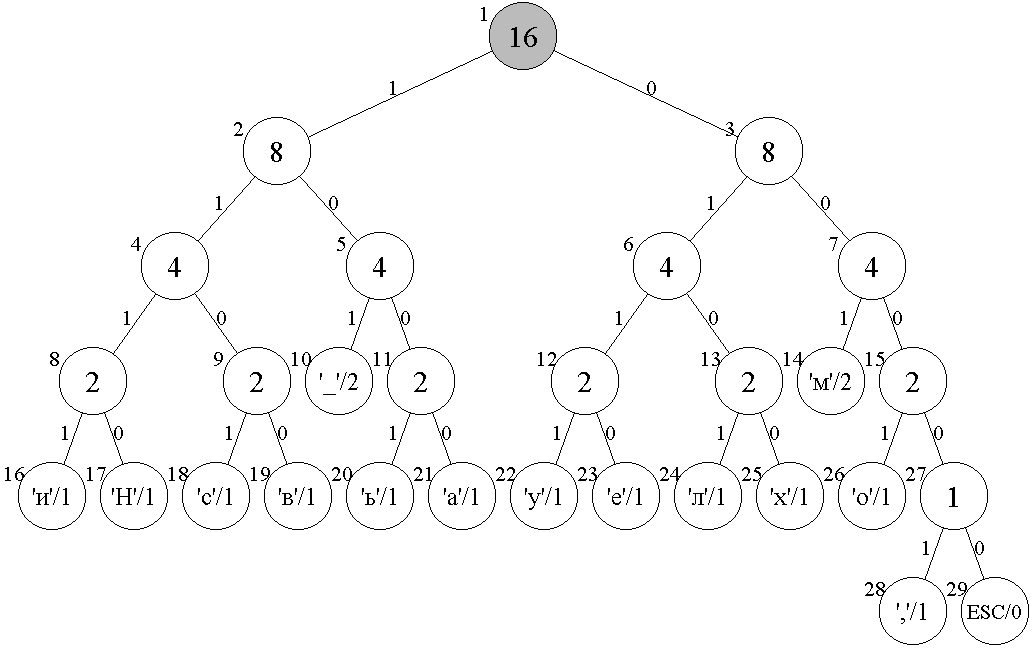


Рисунок 17 – Шестнадцатый этап построения.

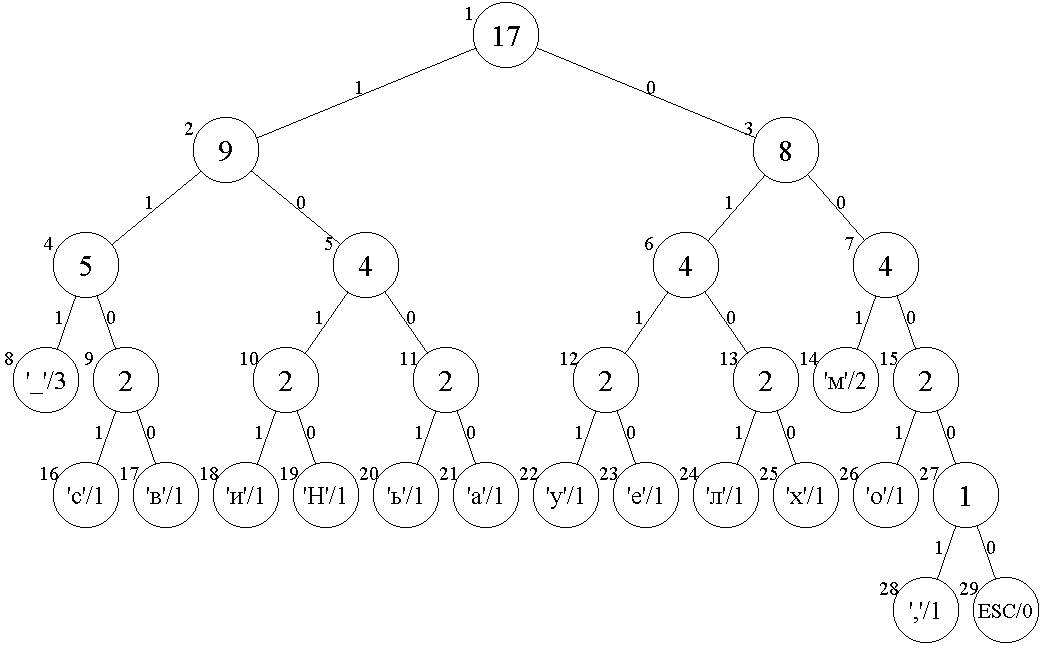


Рисунок 18 – Семнадцатый этап построения.

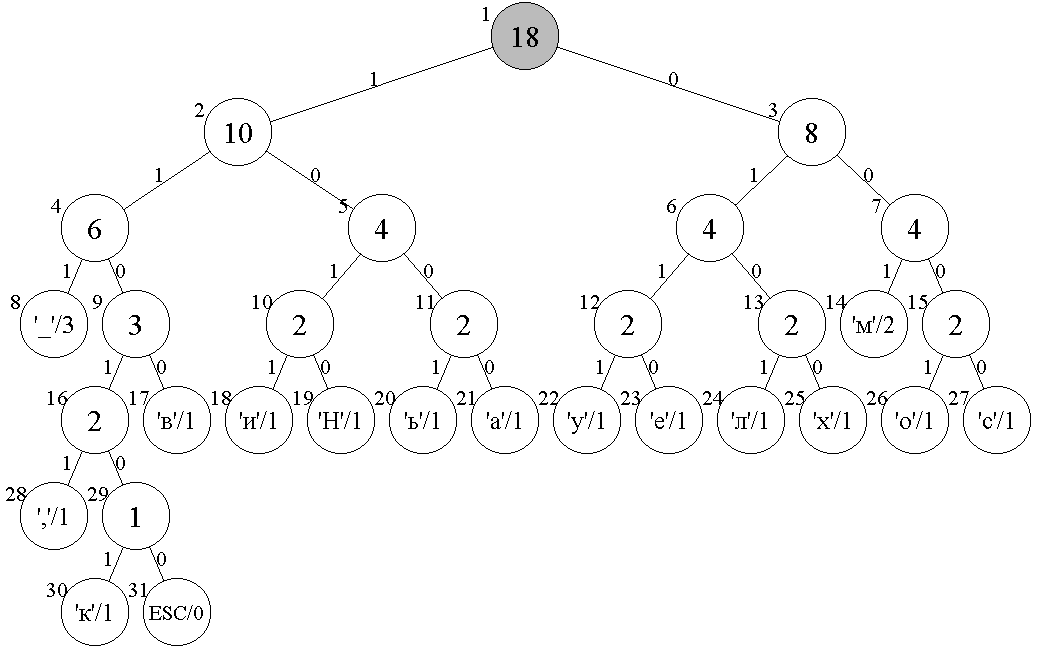


Рисунок 19 – Восемнадцатый этап построения.

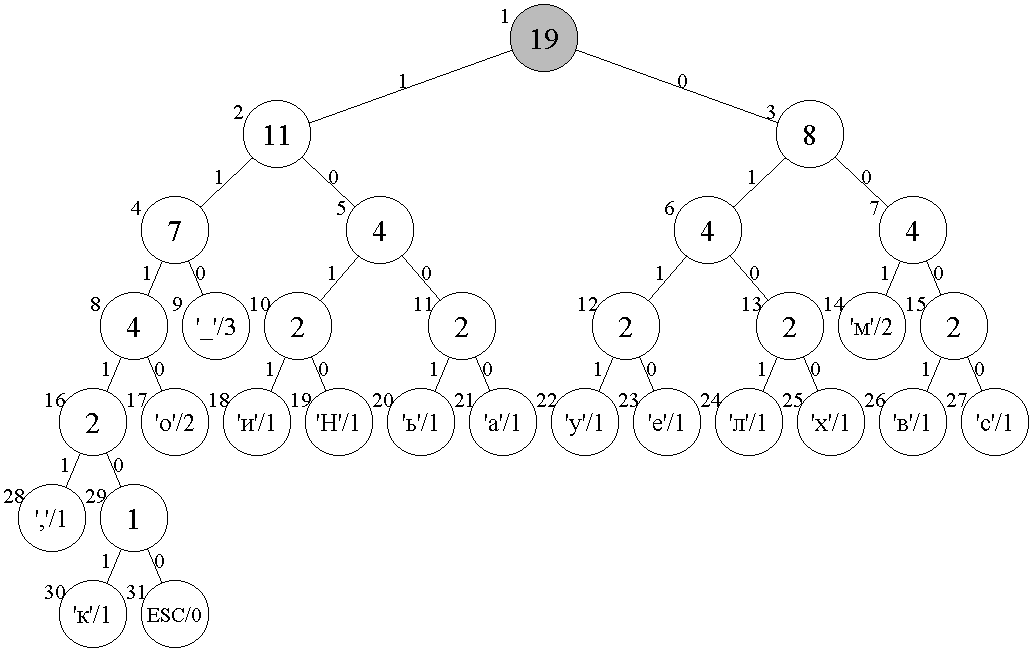


Рисунок 20 – Девятнадцатый этап построения.

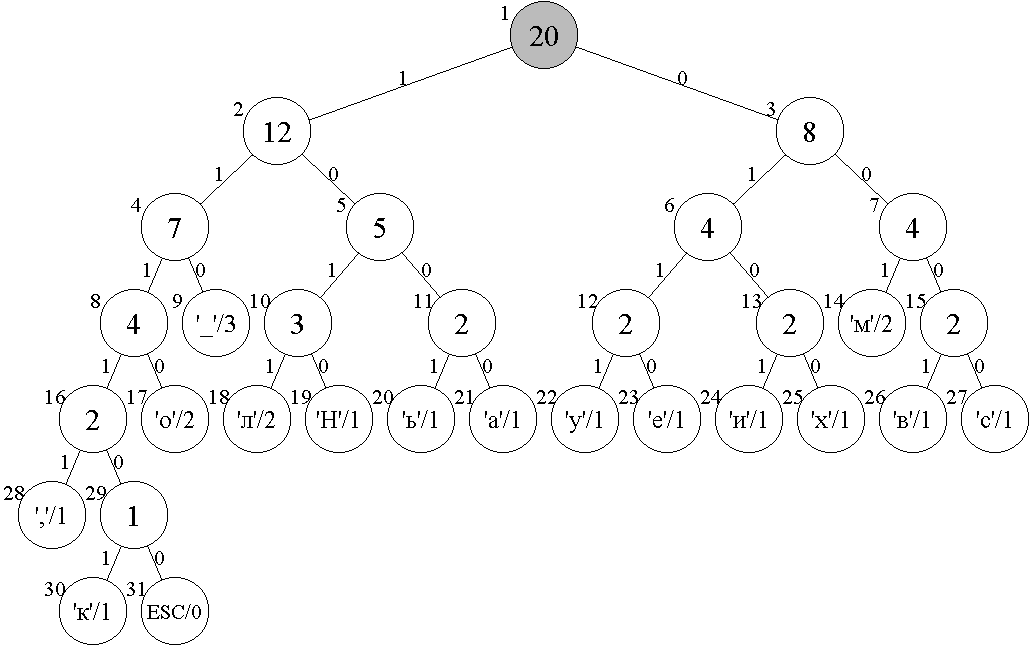


Рисунок 21 – Двадцатый этап построения.

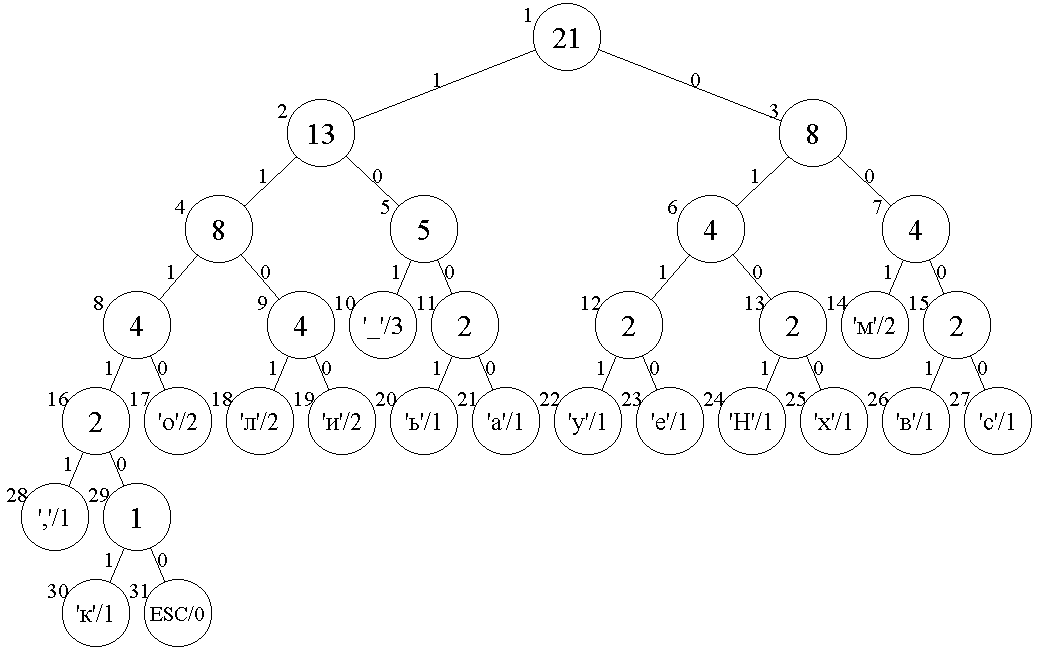


Рисунок 23 – Двадцать первый этап построения.

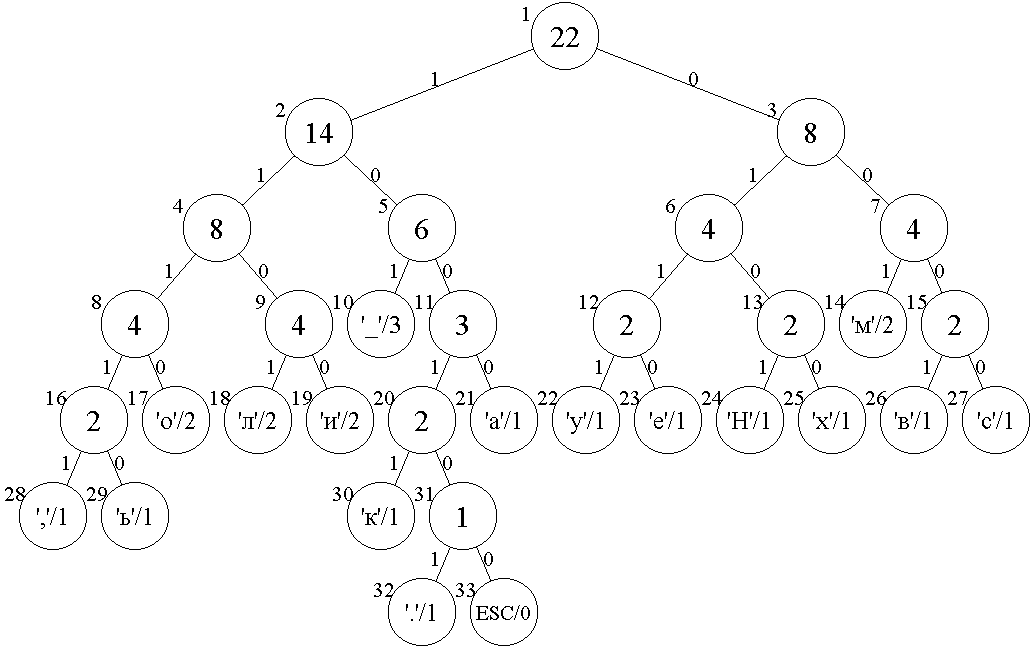


Рисунок 24 – Двадцать второй этап построения.

Дерево постоено. Составим таблицу с полученными кодировками символов.

|  |  |
| --- | --- |
| Литера | Код символа |
| Н | 0101 |
| е | 0110 |
| \_ | 101 |
| х | 0100 |
| в | 0001 |
| а | 1000 |
| л | 1101 |
| и | 1111 |
| с | 0000 |
| ь | 11110 |
| у | 0111 |
| м | 001 |
| о | 1110 |
| , | 11111 |
| к | 10011 |
| . | 100101 |

Таблица 2 – Код символов по дереву Хаффмана.

Выходное сообщение: 0101 0110 101 0100 0001 1000 1101 1111 0000 11110 101 0111 001 1110 001 11111 101 10011 1110 1101 1111 100101

Количество символов = 22; длина сообщения = 88 бит - 35/53 – («0»/«1»)

Средняя длина символа в сообщении = 88/22 = 4 бит/символ.

Коэффициент сжатия:

1. Декодирование сообщения, закодированного методом Хаффмана.

Для декодирования можно воспользоваться построенным деревом. Начинаем с корня дерева и в качестве текущего бита берем начало текста. Смотрим, какое значение у текущего бита, и идем в низ по дереву по дуге, на которой указано такое же значение. Переходим к следующему биту. Если сейчас находимся в листе дерева: читаем символ находящийся в этом листе и записываем его в результат декодирования, переходим снова в корень дерева.

|  |  |
| --- | --- |
| Код | Символ |
| 11001 | Х |
| 11010 | л |
| 001 | е |
| 10110 | б |
| 111 | \_ |
| 000 | д |
| 010 | а |
| 10011 | в |
| 11000 | о |
| 10100 | - |
| 0111 | к |
| 10010 | р |
| 1000 | с |
| 10111 | т |
| 10101 | ь |
| 0110 | я |
| 110111 | н |
| 1101101 | . |

Таблица 3 – Декодирование по дереву Хаффмана.

Полученное сообщение: Не\_хвались\_умом,\_коли.

1. Кодирование сообщения методом LZ77.

Длина словаря = 15 бит.

Длина буфера = 7 бит.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Скользящее окно | | | Выходной код | | |
| Словарь | Буфер | Соврад. Фраза | i | j | s |
|  | Не\_хвал |  | 1 | 0 | Н |
| Н | е\_хвали |  | 1 | 0 | е |
| Не | \_хвалис |  | 1 | 0 | \_ |
| Не\_ | хвались |  | 1 | 0 | х |
| Не\_х | вались\_ |  | 1 | 0 | в |
| Не\_хв | ались\_у |  | 1 | 0 | а |
| Не\_хва | лись\_ум |  | 1 | 0 | л |
| Не\_хвал | ись\_умо |  | 1 | 0 | и |
| Не\_хвали | сь\_умом |  | 1 | 0 | с |
| Не\_хвалис | ь\_умом, |  | 1 | 0 | ь |
| Не\_хвались | \_умом,\_ | \_ | 8 | 1 | у |
| Не\_хвались\_у | мом,\_ко |  | 1 | 0 | м |
| Не\_хвались\_ум | ом,\_кол |  | 1 | 0 | о |
| Не\_хвались\_умо | м,\_коли | м | 2 | 1 | , |
| е\_хвались\_умом, | \_коли. | \_ | 6 | 1 | к |
| хвались\_умом,\_к | оли. | о | 5 | 1 | л |
| ались\_умом,\_кол | и. | и | 12 | 1 | . |

Таблица 4 – Кодирование методом LZ77.

Результат кодирования: 10Н 10е 10\_ 10х 10в 10а 10л 10и 10с 10с 10ь 81у 10м 10о 21, 61к 51л 12 1.

Длина (i) = log212 = 3,585 = 4 (бит)

Длина (j) = log21 = 0 (бит)

Длина исходного сообщения = 88 (бит)

Длина символа ASCII = 4 (бит)

Длина закодированного сообщения = 17\*(4+0+4) = 136 (бит)

Средняя длина символа в сообщении = 136/22 = 7 бит/символ.

Коэффициент сжатия:

1. Декодирование сообщения, закодированного кодом LZ

Процесс декодирования аналогичен кодированию, за исключением того, что нет необходимости искать совпадающие подстроки в словаре, т.к. смещение и длина совпадения уже известны.

Входное сообщение: 10Н 10е 10\_ 10х 10в 10а 10л 10и 10с 10с 10ь 81у 10м 10о 21, 61к 51л 12 1.

|  |  |  |
| --- | --- | --- |
| Словарь | Входные символы | Выходные символы |
|  | 10Н | Н |
| Н | 10е | е |
| Не | 10\_ | \_ |
| Не\_ | 10х | х |
| Не\_х | 10в | в |
| Не\_хв | 10а | а |
| Не\_хва | 10л | л |
| Не\_хвал | 10и | и |
| Не\_хвали | 10с | с |
| Не\_хвалис | 10ь | ь |
| Не\_хвались | 81у | \_у |
| Не\_хвались\_у | 10м | м |
| Не\_хвались\_ум | 10о | о |
| Не\_хвались\_умо | 21, | м, |
| е\_хвались\_умом, | 61к | \_к |
| хвались\_умом,\_к | 51л | ол |
| ались\_умом,\_кол | 12 1. | и. |

Таблица 4 – Декодирование сообщения.

Выходное сообщение: Не\_хвались\_умом,\_коли.

1. Кодирование сообщения методом LZW.

При этом методе кодирования последовательно считывают символы входного потока (строку) и проверяют, есть ли в уже созданной таблице такая строка; если строка есть, то считывают следующий символ. Если такой строки нет, то в выходной поток заносят код для предыдущей найденной строки. Строку

|  |  |  |
| --- | --- | --- |
|  | Символ | Код.слово |
| 0 | \_ | 000000 |
| 1 | л | 000001 |
| 2 | м | 000010 |
| 3 | о | 000011 |
| 4 | Н | 000100 |
| 5 | е | 000101 |
| 6 | х | 000110 |
| 7 | в | 000111 |
| 8 | а | 001000 |
| 9 | и | 001001 |
| 10 | с | 001010 |
| 11 | ь | 001011 |
| 12 | у | 001100 |
| 13 | , | 001101 |
| 14 | к | 001110 |
| 15 | . | 001111 |
| 16 | Не | 010000 |
| 17 | е\_ | 010001 |
| 18 | \_х | 010010 |
| 19 | хв | 010011 |
| 20 | ва | 010100 |
| 21 | ал | 010101 |
| 22 | ли | 010110 |
| 23 | ис | 010111 |
| 24 | сь | 011000 |
| 25 | ь\_ | 011001 |
| 26 | \_у | 011010 |
| 27 | ум | 011011 |
| 28 | мо | 011100 |
| 29 | м, | 011101 |
| 30 | ,\_ | 011110 |
| 31 | \_к | 011111 |
| 32 | ко | 100000 |
| 33 | ол | 100001 |
| 34 | и. | 100010 |

заносят в таблицу и начинают поиск снова. Так как в нашем словаре уже есть 18 символом с кодами 0-17, то кодирование начинаем с 18.

|  |  |  |
| --- | --- | --- |
| Входные символы | Входной код | Новые символы словаря |
| Не | Н | 16=Не |
| \_ | е | 17=е\_ |
| х | \_ | 18\_х |
| в | х | 19=хв |
| а | в | 20=ва |
| л | а | 21=ал |
| и | л | 22=ли |
| с | и | 23=ис |
| ь | с | 24=сь |
| \_ | ь | 25=ь\_ |
| у | \_ | 26=\_у |
| м | у | 27=ум |
| о | м | 28=мо |
| м | о | 29=ом |
| , | м | 30=м, |
| \_ | , | 31=,\_ |
| к | \_ | 32=\_к |
| о | к | 33=ко |
| л | о | 34=ол |
| и. | <22> | 35=и. |
| <EOF> | . |  |

Таблица 6 – Кодирование методом LZW.

Закодированное сообщение: Не\_хвались\_умом,\_ко<22>.

Длина закодированного сообщения = 19\*6 = 114 (бит)

Длина исходного сообщения = 88 (бит)

Средняя длина символа в сообщении = 114/22 = 5,182 = 6 (бит/символ)

Коэффициент сжатия:

1. Декодирование сообщения, закодированного методом LZW.

Вход: Не\_хвались\_умом,\_ко<22>.

|  |  |  |
| --- | --- | --- |
| Входные символы | Выходные символы | Новые символы словаря |
| Н | Не |  |
| е | \_ | 16=Не |
| \_ | х | 17=е\_ |
| х | в | 18\_х |
| в | а | 19=хв |
| а | л | 20=ва |
| л | и | 21=ал |
| и | с | 22=ли |
| с | ь | 23=ис |
| ь | \_ | 24=сь |
| \_ | у | 25=ь\_ |
| у | м | 26=\_у |
| м | о | 27=ум |
| о | м | 28=мо |
| м | , | 29=ом |
| , | \_ | 30=м, |
| \_ | к | 31=,\_ |
| к | о | 32=\_к |
| о | л | 33=ко |
| <22> | и. | 34=ол |
| . |  | 35=и. |

Таблица 7 – Декодирование сообщения.

Выходное сообщение: Не\_хвались\_умом,\_коли.

1. Составить пример черно-белого бинарного изображения размером 8х8 пикселей и закодировать его по методу RLE.

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

Рисунок 25 – Исходное двоичное изображение размеров 8 х 8 пикселей.

Просканируем изображение по строкам. Результат запишем в виде вектора.

|  |
| --- |
| Х=( 00000000 00101001 00101101 00101011 00101001 11101001 01001001 00000000) |

Длина Х = 64 бит. Скорость исходного кода = 1 (бит на элемент изображения).

Выделим участки, на которых данные сохраняют неизменное значение, и определим их длины. Запишем результирующую последовательность длин участков в виде:

r=( 10, 1, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 4, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 8)

Теперь эту последовательность закодируем статическим кодом Шеннона-Фано:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Символы | Кодовые слова ДвК | Частоты символов | 1 деление | 2 деление | 3 деление | 4 деление | Кодовые слова КШФ |
| 1 | 0001 | 22/35 | 0 |  |  |  | 0 |
| 2 | 0010 | 10/35 | 1 | 0 |  |  | 10 |
| 4 | 0100 | 1/35 | 1 | 1 | 0 |  | 110 |
| 8 | 1000 | 1/35 | 1 | 1 | 1 | 0 | 1110 |
| 10 | 1010 | 1/35 | 1 | 1 | 1 | 1 | 1111 |

Таблица 7 – Кодирование статическим кодом Шеннона-Фано.

Т.к. кодируемая последовательность начинается с нуля, не будем ничего добавлять. В результате получим кодовое слово:

B(r) = (11110001001000100010000010100001011000100001001001110)

Длина закодированного слова = 53 бит.

Результирующая скорость - 53/64 = 0,828 ( бит на элемент изображения).

ВЫВОДЫ

В ходе выполнения данной лабораторной работы были исследованы динамические методы сжатия данных: Хаффмана, Лемпеля-Зива (LZ), Лемпеля-Зива-Велча (LZW), RLE.

Было выяснено, что в методе LZW декодер, как и при кодировании, добавляет новые строки в словарь всякий раз, когда находит во входном потоке новый код. В конце процесса декодирования декодер имеет точно такой же словарь новых символов, какой в процессе кодирования был накоплен кодером, при этом его создание было частью процесса декодирования. Т.е. словарь новых символов составляется самим декодером (без передачи).

Методы LZ и LZW привели к увеличению длины сообщения (Kсж. = 1,375 для LZ и Kсж. = 1,125 для LZW). Эти методы дадут хороший результат при условии, если исходное сообщение будет состоять из большего количества совпадающих последовательностей символов.

Метод RLE кодирует изображение эффективно в случае, если оно состоит из малого количества непрерывных участков, на которых данные сохраняют неизменное значение. В обратном случае он может также приводить к увеличению длины закодированного изображения.