Отчет “Кодирование Хаффмана”

БИБ192

Сахар А.С.

2019

**Функциональные требования к программе:**

1) программа должна принимать на вход произвольную последовательность символов, вводимую пользователем;  
2) строить код Хаффмана для входной последовательности символов и выводить построенный код (в виде таблицы);  
3) сжимать входную последовательность с помощью построенного кода Хаффмана и выводить результат пользователю.

Для проверки работы данной программы необходимо создать тестовую выборку (2-3 исходные строки, с уже рассчитанными “вручную” кодами Хаффмана)

**Теория:**

Строка, закодированная методом Хаффмана, использует код, который удовлетворяет условию Фано (ни один код символа не является началом какого-либо другого кода), поэтому может быть однозначно декодирована при наличии таблицы кода Хаффмана или таблицы встречаемости символов в исходном тексте. Сжатие строки осуществляется за счет неравномерности используемых кодов. Длинна кода для символа пропорциональна величине , где — частота вхождения символа в исходную строку.

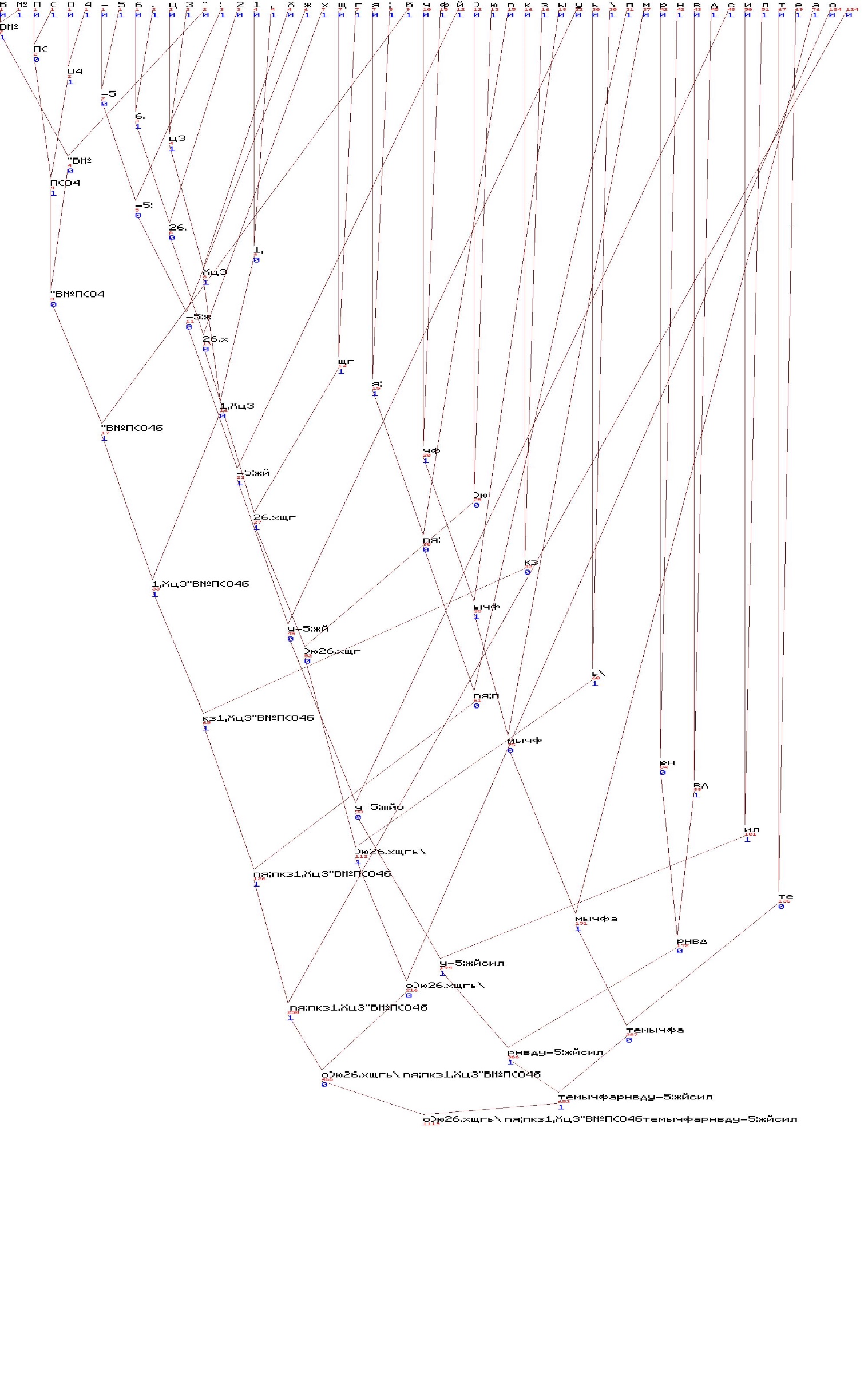
Код Хаффмана строится следующим образом:

1. Рассчитывается таблица встречаемости символов для исходной строки.
2. Из этой таблицы берутся два (две) символа (последовательности символов) с наименьшей частотой встречаемости.
3. Их частоты суммируют и значение записывают в таблицу встречаемости напротив конкатенированной строки из этих символов (последовательностей символов).
4. В выходную таблицу к коду каждого из взятых символов приписывается слева ‘0’ или ‘1’, в зависимости от их частоты встречаемости (первый ‘0’, второй ‘1’ — если частота первого меньше или равна частоте второго; первый ‘1’, второй ‘0’ — в обратном случае)
5. Действия 2-4 производят пока количество строк в таблице встречаемости не станет равным единице.

**Тестовые примеры:**

Строка 1:

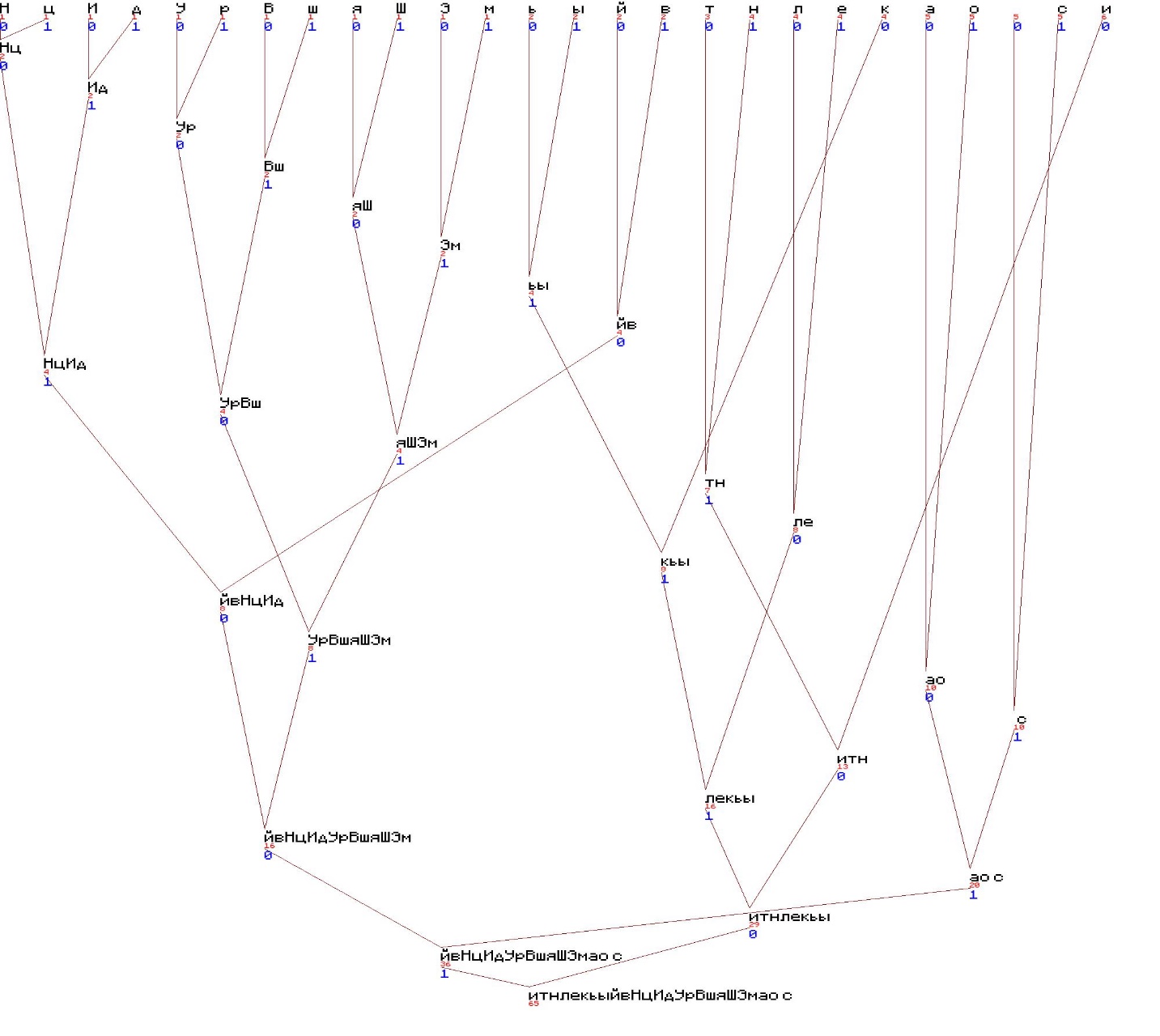
*“В рамках практической работы № 1 необходимо выполнить следующее задание:\\n1) написать программу, осуществляющую сжатие вводимой пользователем последовательности символов по методу Хаффмана;\\n2) подготовить отчет о выполнении практической работы.\\n\\nПрограмма должна обладать следующей функциональностью:\\n1) принимать на вход произвольную последовательность символов, вводимую пользователем;\\n2) строить код Хаффмана для входной последовательности символов и выводить построенный код (в виде таблицы);\\n3) сжимать входную последовательность с помощью построенного кода Хаффмана и выводить результат пользователю.\\n\\nОтчет должен содержать следующие составные части:\\n1) титульный лист с указанием темы работы, фамилии студента и номера группы;\\n2) раздел с заданием на практическую работу;\\n3) раздел с краткой теоретической частью;\\n4) раздел с двумя-тремя примерами "ручного" построения кода Хаффмана для произвольных последовательностей символов;\\n5) раздел с результатами работы программы для тех же последовательностей символов, что и в предыдущем разделе;\\n6) раздел с выводами о проделанной работе” – 1119 символов*



Строка 2:

*“Национальный Исследовательский Университет Высшая Школа Экономики” — 65 символов*

**

**

**Результаты работы программы:**

В ответе сначала идет словарь кодов Хаффмана, в котором каждому символу сопоставлен код в двоичном представлении, а за ним последовательность байт, в которой первый байт — число n от 0 до 7, которое обозначает количество нулей, дописанных слева к двоичному числу, для записи его в виде байт-строки. Далее следует сама байт-строка, в двоичном представлении которой первые n нулей не несут никакой информации.

1. [Пример1](Пример1%20-%20Jupyter%20Notebook.html)

{'В': '0111110010',

' ': '010',

'р': '11000',

'а': '1011',

'м': '10100',

'к': '011100',

'х': '0010101',

'п': '01101',

'т': '1000',

'и': '11110',

'ч': '1010110',

'е': '1001',

'с': '11101',

'о': '000',

'й': '1110011',

'б': '0111111',

'ы': '101010',

'№': '0111110011',

'1': '01111000',

'н': '11001',

'д': '11011',

'в': '11010',

'л': '11111',

'ь': '00110',

'у': '111000',

'ю': '001001',

'щ': '0010110',

'з': '011101',

':': '111001001',

'\': '00111',

'n': '011000',

')': '001000',

'г': '0010111',

',': '01111001',

'я': '0110010',

'ж': '11100101',

'Х': '01111010',

'ф': '1010111',

';': '0110011',

'2': '00101000',

'.': '001010011',

'П': '0111110100',

'ц': '011110110',

'(': '0111110101',

'3': '011110111',

'О': '0111110110',

'4': '0111110111',

'-': '1110010000',

'"': '011111000',

'5': '1110010001',

'6': '001010010'}

b'\x00|\x96.\x8eYT\xdc[\x91\xea\xd3\xd7\x075\x8b~"\xa4\xf9\xa7\x85\x99\x0f\xca\x8d\xfa\x81j\xa6\x8f\xe7\xd0e\xdf\xce\xf8$\xb4\xcav\xf7y\xf4\xf2Nv\x1e\x08Y\xb6\xfb\xb7\x06M\xc0/\x8b\xa58yC\xbc\x16\x9e\xc6\xbe\xc8\x92\xdc\x12\xbb\xcbq\xe9Z\xd0\xdf\xa89\xa6\x8f\x99\xd1\xab\x89\xfc\xd14w\xf3\xb1\xab\x89\xf9\xb2;\x1e]\xf54>4\x9a\x15&\x06\xf8OW^\xbd/6\xcc\xe7`\xa0\x84\xd1\xb2\xe2\x06\xbd\x06B+L \xb5SG\xf39\xf7\x93qnG\xabO\\\x1c\xd6-\xf8\x8a\x8ag;\x07;\x0f\xa6\x01|])kc\xfc\xb9\xb4\x1f\xfe\xf7pe\xdf\xce\xf8$\xb4\xf3U\xf8\xcb\x8fo\x0c\xdf\xcd\x91\xd81<\x93\x9d\x87\x82\x13q\xec\xfa\x97\x06Y\xb5\xa2\xa3i\xb8\x1ewC\xe6\xcf\x04\xa6\x8e\xfev5q?6G`\xcb\xbe\xa6\x87\xc6\x9eV\xb47\xeapJh\xf9\x9d\x1a\xb8\x9f\xcd\x19\x9c\xec\x14\x10\xbb\x18\x1e\x83\'\x06\xd3\xd5\xd7\xafK\xcd\xad\xfd\x92\xd1Q\xbc\x8ei\xa3\xbf\x9d\x8d\\O\xcd\x91\xd8\xf2\xef\xa9\xa1\xf1\xa5\xe5\xaa\xb47\xe82h\xec`NsW4\xe0\xda}t\xb5\xed\xca\x8b\x7f\xff=\xaa!\x99\xce\xc3\xdc\x85\xde\_R\xe0\xcbEF\xf3\xc1)\xa3\xbf\x9d\x8d\\O\xcd\x91\xd82\xeah\xa0,bSGc\x02s\x90\\\'\x06\xed=]z\xf4\xbc\xda\xf2\xd5Z\x1b\xf4\x19b]\xe3\xe6\x8b\x84\xd1\xf3:5q?$\xa6s\xb0s\xb0\xfbEi\x85\xb1\xfeY\xca\xe8\xdc\xe3\x96\xe0\xcb\xbf\x9d\xf0IoJ\xe8\xec^\xb3T\xaa\xd7\xd8\xf7$\xe7a\xe0\x85\x1e\x8e>l\xd5\xcd\x7f\xdd\x85\xd5\xc3\x96\xed\xe7\xd3E\x13J\x96-\xf8\x8a\x9eU{\xa7\xbf\xef.\xc7\x1b\x9c\xc5\xaf,\x8aN-\x17\xc7\rmL\xcev\n\x08X\xb7w?]N\xde\xef>\x9a,\xdan-\xc8\xf5i\xeb\x9c\x12\xb1o\xc4p\xcc\xe7a\xeeB\xc5\xbb\xb9\xfa\xeas\x17\x0e\x0ej$bc\xd5\xa7\xae\x0ej\xb5\xf6\x0cK3\x9d\x87\xdc\x85\x8bws\xf5\xd5\xbdqF\\\x84bh\xc97\x1e\xa4\xe2\xe9\xe4\xf8\xc7\x15\xb2\x0b\x87\xc2h\xec`N|\xc987i\xea\xeb\xd7\xa5\xe6\xd6\xfe\xc97\x03\xce\xe8|\xd9\xa8\xaah\xef\xe7cW\x13\xf3dv\'\x9a\xef\xa9\xa1\xf1\xa6g;\x1c\x89\x0b\x16\xee\xe7\xeb\xab\x12\xef\x1f4\\]<\xb1o\xc4T\x9b\x80\_\x17JT\xb7\xf6J$\xaa\xe5\x94\xd1\xdf\xce\xc6\xae\'\xe6\xc8\xecO5\xdfSC\xe3O\*\xb4\x05\xe5\xa4\xdcN\xea\xdf\x05\xa6\x8b\x16\xee\xe7\xf2\xcc\xe7`\xa4B\xc5\xbb\xb9\xfa\xea\xd5Z\x1b\xbay\x04\xdc\r\xcf\xef9\x1c\xd6-\xf8\x89'

Длинна сжатой строки: 669

1. [Пример2](Пример2%20-%20Jupyter%20Notebook.html)

{'Н': '100100',

'а': '1100',

'ц': '100101',

'и': '000',

'о': '1101',

'н': '0011',

'л': '0100',

'ь': '01110',

'ы': '01111',

'й': '10000',

' ': '1110',

'И': '100110',

'с': '1111',

'е': '0101',

'д': '100111',

'в': '10001',

'т': '0010',

'к': '0110',

'У': '101000',

'р': '101001',

'В': '101010',

'ш': '101011',

'я': '101100',

'Ш': '101101',

'Э': '101110',

'м': '101111'}

b'\x02$\xc9F\x9e#\x8d\xf0\xe9\xbf\xd1g\xd8\xe1\*;\xd8C\xa81\x15\xa7\xc1)uO\xfa\xf2\xce\xb5\xb53\xaem=\xbc'

Длинна сжатой строки: 36 байт

**Выводы:**

1. Программа работает относительно неэффективно по времени и используемой оперативной памяти из-за применения строковых переменных для записи двоичных кодов символов (произведена проверка на больших объемах данных, 2Гб).
2. Неадаптивным методом Хаффмана можно эффективно сжимать строки относительно большого объема при большой частоте встречаемости отдельных символов.
3. При большом разнообразии и относительно небольшой длине строки сжатие таким методом происходит не эффективно.