Отчет “Адаптивное кодирование Хаффмана”

БИБ192

Сахар А.С.

2019

**Функциональные требования к программе:**

1) программа должна принимать на вход произвольную последовательность символов, вводимую пользователем;  
2) строить адаптивный код Хаффмана для входной последовательности символов;  
3) выводить результирующий код Хаффмана, полученный после обработки всех символов входной последовательности;  
4) сжимать входную последовательность с помощью адаптивного кодирования Хаффмана и выводить результат пользователю;  
5) выполнять декодирование сжатой последовательности.

Для проверки работы данной программы необходимо создать тестовую выборку (2-3 исходные строки, с уже построенным “вручную” адаптивным кодом 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.

В отличие от полу-адаптивного кодирования Хаффмана, при построении адаптивного кода можно использовать потоковые данные, так как дерево перестраивается по мере прохождения алгоритма по исходному тексту.

Для кодирования символов, которые ранее не были встречены в тексте на дерево добавляется esc-символ. Его код добавляется в выходной поток перед передачей символа строки в несжатом виде.

**Тестовые примеры:**

Строка 1:

*“Адаптивное кодирование Хаффмана” – 62Байта (кодировка UTF-16)*

Шаги ручного построения находятся в папке “sample1steps”

Cтрока 2:

*“Национальный Исследовательский Университет Высшая Школа Экономики” — 130Байт (кодировка UTF-16)*

Шаги ручного построения находятся в папке “sample2steps”

**Результаты работы программы:**

В ответе сначала идет результирующая таблица кодов Хаффмана, в котором каждому символу сопоставлен код в двоичном представлении, а за ней 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. esc: 01000

А: 01001

д: 0101

а: 011

п: 11000

т: 11001

и: 1111

в: 1000

н: 000

о: 001

е: 1001

: 1010

к: 11010

р: 11011

Х: 11100

ф: 1011

м: 11101

b'\x02\x00\x82\x00CH\x10\xc0\x04?\xc0\x88P\x10\xe1\x02\x19\x00C\xde\x04>\xc0CZ\x00 \x80C\xaf\x8b\x00D\x05(\xde\x89@B[0" \x88\x10\xf0K'

Длинна сжатой строки: 47 байт

Decompressed message: 'Адаптивное кодирование Хаффмана'

1. esc: 1011110

Н: 1011111

а: 1100

ц: 100100

и: 000

о: 1101

н: 0011

л: 0100

ь: 01110

ы: 01111

й: 10000

: 1110

И: 100101

с: 1111

е: 0101

д: 100110

в: 10001

т: 0010

к: 0110

У: 100111

р: 101000

В: 101001

ш: 101010

я: 101011

Ш: 101100

Э: 101101

м: 101110 b'\x02\x00\x83\xa0C\x08\x11\x18\x048\xc0\x87\xd0\x10\xf6\x00C\xbe\x04Lt\x08\x97\x00\x87,\x00@\x80\x83\x04\x08\x83\xf3\xc0C]\x02\x1a%\x81\x0c\xae\x80\x88G|\xb1\x02\x1d\_\x18\xe8\x10\x8dN\x8dP"\x07\\5M\xe0A#\xeb@\x89\r@"|\x1c\x08Q\xfe\x19\xe7\xb0!m\xc8\x99\x00C\xc4='

Длинна сжатой строки: 88 байт

Decompressed message: 'Национальный Исследовательский Университет Высшая Школа Экономики'

**Выводы:**

1. Строка, сжатая адаптивным алгоритмом Хаффмана, занимает меньше памяти чем исходная.
2. При помощи написанной программы сжатую строку можно декодировать обратно.
3. Для обработки потоковых данных в функции сжатия необходимо добавить мемориализацию.