Метод "стопка книг"

5 января 2023 г. 1:25

Имеем алфавит источника $X = \{0, 1, \dots, M-1\}$ и последовательность $x_1, x_2, \dots$ на его выходе. Кодер и декодер предполагает, что первой букве предшествуют все остальные буквы алфавита.
В таком случае можно применить интервальное кодирование. В нем мы предполагаем, что на начало выполнения алгоритма имеем строку - последовательность всех букв алфавита. И для каждой буквы сообщения ищем количество букв между ней и предыдущей такой же буквой - это и будет кодовое слово.

$abc|cabbbabbac \to 0, 3, 3, 0, 0, 3, 1, 0, 2, 9$

Или можно применить **"стопку книг"**. Этот алгоритм учитывает повторения, что улучшает сжатие. Вместо количества букв до предыдущей такой же "стопка книг" считает количество **различных** букв до предыдущей такой же.

$abc|cabbbabbac \to 0, 2, 2, 0, 0, 1, 1, 0, 1, 2$

$cba|cabbbabbac \to 2, 1, 2, 0, 0, 1, 1, 0, 1, 2.$

Декодер, соответственно, знает кол-во различных букв до предыдущей такой же, идет с конца считать разные буквы, и как только счетчик достигает значения на 1 больше, становится ясно, какая буква была закодирована.

Итак, в результате кодирования получается некоторая последовательность чисел. Далее к ним можно применить простой побуквенный код. Это будет не оптимально с точки зрения Хаффмана, например, но так делается в случае, если чем больше число, тем реже оно появляется. А здесь мы подразумеваем повторение символов, часто будут появляться нули - на это мы и опираемся, используя монотонный код.

Монотонный код состоит из двух частей.
Пусть мы применяем монотонный код к числу n.

1. Сначала считаем $log_2(n)$, округляем вниз и прибавляем 1, кодируем полученное унарным кодом. Что это такое - кодируем, сколько бит нам нужно, чтобы закодировать число равномерным кодом
2. Вычисляем $n - 2^{\lfloor log_2(n) \rfloor}$, переводим в двоичную СС, т.е. представляем число в равномерном коде. Здесь получится двоичное число, старший разряд которого всегда 1, можно отбросить 1 в кодере при кодировании, а в декодере подставить обратно

- `IF_WE_CANNOT_DO_AS_WE_WOULD_WE_SHOULD_DO_AS_WE_CAN`
- В ASCII пробел это 32, а буквам A,...,Z соответствуют числа 65...90.
- Результат: ...zyx...ZYX...BA..._...|74, 72, 35, 88, 73, 3, 72, 71, 80, 1, 81, 86, 6, 76, 4, 3, 6, 86, 3, 10, 10, 3, 3, 6, 87, 83, 9, 6, 6, 7, 3, 8, 81, 9, 9, 9, 9, 7, 2, 5, 3, 10, 8, 3, 10, 10, 3, 13, 6, 13.
- Применим к $\{d_i\}$ монотонный код:

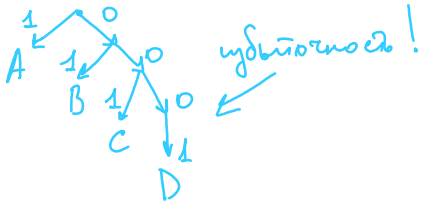
$$mon(n) \to \underbrace{unar\,(\lfloor \log_2 n \rfloor + 1)}_{\underbrace{111..0}} \underbrace{bin(n - 2^{\lfloor \log_2 n \rfloor})}_{\lfloor \log_2 n \rfloor \ бит}.$$

$$mon(21) \to \underbrace{11110}\underbrace{10101}.$$

- Получим 332 бита.

Унарная бинаризация

n	$unar(n)$
1	1
2	01
3	001
4	0001
5	00001
n	$\underbrace{00...0}_{n-1}1$



Унарная бинаризация:

$A \to 1$
$B \to 01$
$C \to 001$
$D \to 0001$

$h_1 = -p_A \log p_A - (1 - p_A) \log(1 - p_A).$
$h_2 = -\frac{p_B}{p_B+p_C+p_D} \log \frac{p_B}{p_B+p_C+p_D} - \frac{p_C+p_D}{p_B+p_C+p_D} \log \frac{p_C+p_D}{p_B+p_C+p_D}.$
$h_3 = -\frac{p_C}{p_C+p_D} \log \frac{p_C}{p_C+p_D} - \frac{p_D}{p_C+p_D} \log \frac{p_D}{p_C+p_D}.$
$h_4 = 0.$

$$h_1 + (1 - p_A)h_2 + (1 - p_A - p_B)h_3 =$$
$$-p_A \log p_A - (1 - p_A) \log(1 - p_A) -$$
$$-p_B \log \frac{p_B}{p_B+p_C+p_D} - (p_C + p_D) \log \frac{p_C+p_D}{p_B+p_C+p_D} -$$
$$-p_C \log \frac{p_C}{p_C+p_D} - p_D \log \frac{p_D}{p_C+p_D} = H.$$
