Реализация двоичного арифметического кодирования без умножений

5 января 2023 г. 1:23

У обычной реализации арифметического кодирования есть проблема: наличие действий умножения и деления. Понятно, что данные операции на разных платформах могут выполняться по-разному, это может привести к нестыковке кодера и декодера. То есть на одной платформе закодировали, на другой раскодировали, но оригинал не получили из-за этого отличия результатов умножения и деления.

Также отметим, что в алгоритмах сжатия JPEG2000, H.264, H.265, H.266 используется двоичный арифметический кодер. То есть алфавит принимается за {0, 1}.
Алгоритмы двоичного арифметического кодера и декодера выглядят так:

```
T ← R×p(x_t)
R ← R − T
if x_t =1 then
    L ← L+R
    R ← T
end if
call Ренормализация
```

```
T ← R×p(x_t)
R ← R − T
if F < R then
    x_t = 0
else
    L ← L+R
    R ← T
    x_t = 1
end if
call Ренормализация
```

Т.е. здесь мы избавляемся от деления, избавляемся от циклов while

------

R - регистр, в котором хранится G (произведение вероятностей)
T по сути тоже произведение вероятностей хранит ?
L - регистр, в котором хранится F (кумулятивная вероятность)
b - разрядность алгоритма

Однако в алгоритме всё ещё присутствует умножение. От него можно избавиться с помощью аппроксимации.
Для начала нужно разобраться, что происходит с регистром R.
Если обратиться к ренормализации, там присутствует цикл while R < 2^(b-2). Самое большое значение R, при котором будет выполняться ренормализация - это 2^(b-2)-1

```
while R < 2^(b−2) do
    if L ≥ 2^(b−1) then
        WriteOnes(1)
        WriteZeros(bits_to_follow), bits_to_follow ← 0
        L ← L − 2^(b−1)
    else if L < 2^(b−2) then
        WriteZeros(1)
        WriteOnes(bits_to_follow), bits_to_follow ← 0
    else
        bits_to_follow ← bits_to_follow + 1
        L ← L − 2^(b−2)
    end if
    L ← L≪1, R ← R≪1
end while
```

После ренормализации регистр R находится в интервале:

$$\frac{1}{2}2^{b-1} \le R < 2^{b-1}.$$

потому что

Поэтому умножением может быть аппроксимировано следующим образом:

$$T = R \times \hat{p}_t \approx \alpha 2^{b-1} \times \hat{p}_t,$$

выбираем α, предвычисляем значения T для всех вероятностей (α знаем, 2^{b-1} = const, p имеем, например, 64 штуки)

где $\alpha \in [\frac{1}{2}, ..., 1)$. Для улучшения точности M-coder квантует интервал $[\frac{1}{2}2^{b-1}; 2^{b-1})$ равномерно на 4 интервала. Затем, для каждого из четырёх интервалов результат умножения $R \times \hat{p}_s$ помещается в таблицу $TabRangeLPS[s][\Delta]$, состоящую из 64×4 значений.

Таким образом, без умножений двоичное арифметическое кодирование переписывается так:

Реализация M-coder[2]

```
Δ ← (R − 2^(b−2)) ≫ (b − 4)
T ← TabRangeLPS[s][Δ]
R ← R − T
if x_i ≠ MPS then
    L ← L+R
    R ← T
    if s = 0 then
        MPS ←!MPS;
    end if
    s ← TransStateLPS[s]
else
    s ← TransStateMPS[s]
end if
call Renormalization procedure
```

1. Вычисляется дельта (это от 0 до 3)
2. Предвычисляется lookup-таблица, из нее берется уже посчитанное значение T
3. Выполняется тот же алгоритм

[s] - номер вероятности
[Δ] от 0 до 3, соответствует одной из "альфа" (показывает, к какому из 4 интервалов обращаться)