Математические модели обработки сигналов

Тема 6: Формат сжатия изображений JPEG

Лектор: Кривошеин А.В.

Изображение

Изображение — это результат фиксации такого физического явления, как свет (и в более общем случае, ЭМ волн).

Простейшая модель изображения — это двумерная функция $f(x, y)$, где (x, y) — координаты на плоскости, а значение $f(x, y)$ отвечает за яркость в этой точке

Цифровое изображение (двумерное, в серых тонах) — это отображение, вида

$I : \{1, \ldots, N_1\} \times \{1, \ldots, N_2\} \to \mathbb{R}$ или $I \in \mathbb{R}^{N_1 \times N_2}$, или матрица размера $N_1 \times N_2$

Элемент матрицы I называют **пикселем** (англ. pixel).

Диапазон значений пикселей:

либо интервал целых чисел от 0 до 255, то есть 8 бит на пиксель,

либо числа в формате double от 0 до 1 или иначе.

Цветное изображения в цветовом пространстве RGB — это три матрицы размера $N_1 \times N_2$, где каждая из матриц отвечает за насыщенность соответствующего цвета.

ДПФ изображений

Базовые частоты для пространства $\mathbb{C}^{N_1 \times N_2}$ получаются также дискретизацией базовых частот из $L_2[0, 1]^2$. Базовые частоты в $L_2[0, 1]^2$ характеризуются частотным индексом (m_1, m_2), который указывает сколько полных периодов колебаний по одной из переменных укладывается в интервале $[0, 1]$ при фиксированной другой переменной. Например, базовая частота, соответствующая частотному индексу (m_1, m_2) имеет вид

$$e^{2\pi i (m_1 t_1 + m_2 t_2)} = e^{2\pi i m_1 t_1} e^{2\pi i m_2 t_2}.$$

Проводя равномерную дискретизацию базовых частот при $t_i = \frac{n_i}{N_i}$, $n_i = 0, \ \ldots, N_i - 1$, $i = 1, 2$, получим массив значений

$$\mathcal{E}_{m_1,m_2}[n_1, n_2] = e^{2\pi i \frac{m_1 n_1}{N_1}} e^{2\pi i \frac{m_2 n_2}{N_2}}.$$

В силу периодичности частотные индексы достаточно рассматривать для значений $m_i = 0, \ \ldots, N_i - 1, \ i = 1, 2$. Этот набор массивов также образует ортонормированный базис для пространства $\mathbb{C}^{N_1 \times N_2}$.

2D ДПФ сигнала $x \in \mathbb{C}^{N_1 \times N_2}$ имеет вид

$$X[m_1, m_2] := \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x[n_1, n_2] \, \mathcal{E}_{m_1,m_2}[n_1, n_2] = \sum_{n_2=0}^{N_2-1} \left(\sum_{n_1=0}^{N_1-1} x[n_1, n_2] \, e^{-2\pi i \frac{m_1 n_1}{N_1}} \right) e^{-2\pi i \frac{m_2 n_2}{N_2}}, \quad m_i = 0, \ \ldots, N_i - 1, \ i = 1, 2.$$

2D ДПФ действует из $\mathbb{C}^{N_1 \times N_2}$ в $\mathbb{C}^{N_1 \times N_2}$ и является по сути сменой базиса.

Базис из комплексных экспонент сепарабелен, то есть может быть представлен как произведение одномерных комплексных экспонент. А значит 2D ДПФ может быть посчитано с помощью БПФ, строка за строкой, и затем столбец за столбцом.

О сжатии изображений

Хранение изображений в формате BMP (англ. Bitmap Picture) в виде матрицы со значениями пикселей очень затратно. Например, изображение в серых тонах размера 1920×1080 пикселей с 8 битами на пиксель потребует $B = 2\,073\,600$ байт. Возникает необходимость в хранении изображений в сжатом виде.

Например, рассмотрим следующий способ: построим базу всех созданных изображений, и каждому присвоим индекс. Оценочно, общее число цифровых изображений порядка 10 трлн - $10^{13} \sim 2^{39}$. Значит для кодирования индекса одного изображения хватит 40 бит. Но это не лучший способ для хранения изображений, поскольку у каждого декодировщика должна быть такая база.

Наиболее распространенный метод — это сжатие изображения с потерями (англ. lossy compression). Системы сжатия строятся таким образом, чтобы обеспечивать минимальную потерю визуального качества изображения. Например, границы значительно более важны в изображении, чем равномерно залитая цветом область. Соответственно, можно выделить большее число бит на границы, и меньше на равномерные области.

Алгоритм JPEG

Формат JPEG внедрён в 1994, но до сих пор показывает хорошую производительность даже в сравнении с современными разработками, то есть высокие степени сжатия могут быть достигнуты без значительных визуальных потерь.
Цветное изображение закодированное в формате JPEG так, что в среднем на один пиксель приходится 2 бита, с точки зрения нормы в $\mathbb{R}^{N_1} \times \mathbb{R}^{N_2}$ практически неотличимо от формата BMP с 24 битами на пиксель. А уровень сжатия, обеспечивающий $\frac{1}{2}$ бита на пиксель, даёт картинку мало отличимую от оригинала визуально.

Рассмотрим шаги алгоритма сжатия изображений в формата JPEG для изображений в серых тонах.

Для цветных изображений сначала изображение преобразуется в оптимальное цветовое пространство. Известно, что в случае применения цветового пространства **яркость/цветность (YCbCr или YUV)** достигается лучшая степень сжатия. Известно, что глаза более чувствительны к яркости, чем к цвету. Поэтому при хранении, можно экономить на U, V компонентах, например, уменьшая для них частоту дискретизации. Формула перевода из RGB в YUV:

$$Y = 0.299R + 0.587G + 0.14B$$
$$U = -0.169R - 0.331G + 0.5B$$
$$V = 0.5R - 0.419G - 0.81B$$

Кодирование в формате JPEG состоит из следующих этапов:

1. Изображение локально преобразуется с помощью DCT (дискретного косинусного преобразования)
2. Полученные коэффициенты со значениями ниже некоторого порога отбрасываются (thresholding)
3. Оставшиеся коэффициенты кодируются эффективным образом (Huffman code)

Алгоритм JPEG: 1 этап, ДКП

1. Сдвиг среднего уровня яркости — вычитание среднего значения 128 (то есть 256/2) из значений пикселей исходного изображения.
2. Разбиение изображения на блоки размера 8 на 8 пикселей.
3. Применение дискретное косинусное преобразование (ДКП, англ. DCT) к каждому блоку.

ДКП — это вещественный аналог ДПФ.

Коэффициенты ДКП вычисляются по формулам:

$$\mathrm{DCT}[k_1,\, k_2] \;=\; \frac{C[k_1]\, C[k_2]}{\sqrt{2\,N}} \sum_{n_1=0}^{N-1} \left(\sum_{n_2=0}^{N-1} x[n_1,\; n_2] \cos\left(\pi\, \frac{(2\,n_2+1)\,k_2}{2\,N} \right) \right) \cos\left(\pi\, \frac{(2\,n_1+1)\,k_1}{2\,N} \right)$$

$$C[m] \;=\; \begin{cases} \frac{1}{\sqrt{2}} & m = 0 \\ 1 & m \neq 0. \end{cases}$$

Восстановление проводится по формулам:

$$x[n_1,\; n_2] = \frac{1}{\sqrt{2\,N}} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} C[k_1]\, C[k_2]\, \mathrm{DCT}[k_1,\, k_2] \cos\left(\pi\, \frac{(2\,n_1+1)\,k_1}{2\,N} \right) \cos\left(\pi\, \frac{(2\,n_2+1)\,k_2}{2\,N} \right).$$

Алгоритм JPEG: 1 этап, связь ДКП и ДПФ

Для одномерного сигнала $x \in \mathbb{R}^N$, его ДКП можно связать с ДПФ сигнала $z \in \mathbb{R}^{2N}$, определяемого как

$$z[n] = \begin{cases} x[n], & n = 0, \ ..., N-1, \\ 0, & n = N, \ ..., \ 2N-1. \end{cases}$$

ДКП имеет вид: $$\mathrm{DCT}[k] = \sum_{n=0}^{N-1} x[n] \cos\left(\pi \frac{(2n+1)k}{2N}\right), \ k = 0, \ ..., N-1,$$

ДПФ сигнала z имеет вид

$$Z[k] = \sum_{n=0}^{2N-1} z[n] \, e^{-2\pi i \frac{kn}{2N}} = \sum_{n=0}^{N-1} x[n] \, e^{-2\pi i \frac{kn}{2N}} = e^{\pi i \frac{k}{2N}} \sum_{n=0}^{N-1} x[n] \, e^{-2\pi i \frac{k(n+1/2)}{2N}}, \quad k = 0, \ ..., 2N-1,$$

Тогда $\mathrm{DCT}[k]$ можно получить как вещественную часть коэффициентов $Z[k]$ домноженных на комплексную экспоненту

$$\mathrm{DCT}[k] = \mathrm{Re}\left(e^{-\pi i \frac{k}{2N}} Z[n] \right).$$

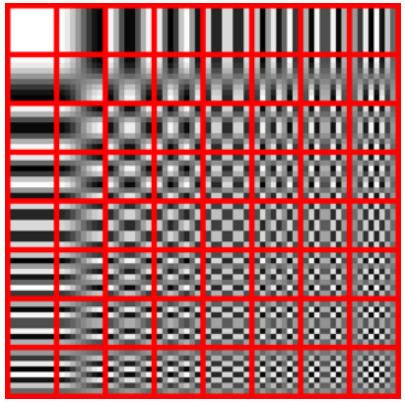
Аналогично, можно установить связь многомерного ДКП с ДПФ. Однако, для ДКП существуют также свои быстрые алгоритмы схожие с БПФ.

Алгоритм JPEG: 1 этап

Смысл каждого коэффициента — это мера похожести блока изображения и базисного блока. Вид 64 базисных блоков

$$w_{k_1,k_2}[n_1, n_2] = \cos\left[\pi \frac{(2n_1+1)k_1}{2N}\right] \cos\left[\pi \frac{(2n_2+1)k_2}{2N}\right], \quad k_1, k_2 = 0, \ldots, N-1,$$

представлен на картинке



Как правило, большинство изображений являются довольно гладкими в рамках блоков 8×8 пикселей. Соответственно, большая часть коэффициентов после ДКП близка к нулю.

Алгоритм JPEG: 2 этап

Второй этап алгоритма JPEG называют **квантованием коэффициентов** ДКП. Этот этап отвечает за степень сжатия. Внутри блока коэффициенты квантуются не одинаково. На основе психо-визуальных опытов, были созданы таблицы для квантования, дающие наиболее предпочтительные результаты с точки зрения качества изображения и степени сжатия.

Пример таблицы для квантования:

$$M = \{m_{n,k}\} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

Формула квантования : $C_q[n, k] = \text{round}\left(\frac{\text{DCT}[n, k]}{m_{n,k}}\right), \quad n, k = 0, \ ..., 7$

Когда значение $\frac{C[n,k]}{m_{n,k}}$ меньше $\frac{1}{2}$, то на выходе получим нулевой коэффициент и многие действительно обнулятся.

Указанная таблица ассоциирована с показателем качества $F_q = 50\ \%$.

Для иных значений качества F_q вычисляется коэффициент scale:

Если $F_q < 50\ \%$, $\text{scale} = \frac{5000}{F_q}$,

Для $F_q > 50\ \%$, $\text{scale} = 200 - 2\,F_q$.

Новая таблица Q_{new} получается из эталонной M в виде

$Q_{\text{new}} = \text{Floor}((M \cdot \text{scale} + 50)/100)$, где Floor округление к ближайшему целому снизу.

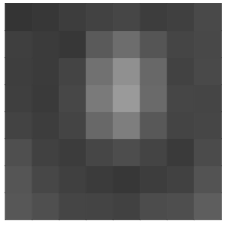
Далее, значения меньше либо равные 1, полагаются равными 1, значения большие 255 полагаются равными 255.

Алгоритм JPEG: 2 этап

Рассмотрим первые 2 этапа на примере:

Out[ ]=

$$\begin{pmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{pmatrix}$$



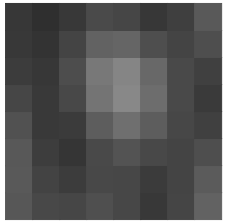
$$\rightarrow \begin{pmatrix} -415.0 & -21.3 & -43.3 & 19.3 & 39.7 & -14.2 & -1.7 & 0.3 \\ 3.2 & -10.9 & -30.4 & 5.1 & 6.6 & -3.5 & -4.3 & 2.4 \\ -33.1 & 3.7 & 38.6 & -12.3 & -14.5 & 5.0 & 2.7 & -2.8 \\ -34.3 & 6.0 & 17.0 & -7.4 & -5.1 & 3.1 & 0.9 & 1.0 \\ 8.6 & -3.3 & -6.6 & -2.0 & -0.9 & 0.9 & -1.4 & 1.6 \\ -5.5 & 1.5 & 1.2 & -3.0 & -1.2 & 0.5 & 2.2 & 0.9 \\ -0.7 & 0.1 & 0.2 & -1.2 & -0.4 & -1.5 & 2.1 & -0.3 \\ -0.1 & 0.1 & -0.5 & -2.1 & -0.6 & -0.0 & 0.3 & 0.8 \end{pmatrix}$$

Квантование (по таблице, указанной выше):

$$\begin{pmatrix} -415.0 & -21.3 & -43.3 & 19.3 & 39.7 & -14.2 & -1.7 & 0.3 \\ 3.2 & -10.9 & -30.4 & 5.1 & 6.6 & -3.5 & -4.3 & 2.4 \\ -33.1 & 3.7 & 38.6 & -12.3 & -14.5 & 5.0 & 2.7 & -2.8 \\ -34.3 & 6.0 & 17.0 & -7.4 & -5.1 & 3.1 & 0.9 & 1.0 \\ 8.6 & -3.3 & -6.6 & -2.0 & -0.9 & 0.9 & -1.4 & 1.6 \\ -5.5 & 1.5 & 1.2 & -3.0 & -1.2 & 0.5 & 2.2 & 0.9 \\ -0.7 & 0.1 & 0.2 & -1.2 & -0.4 & -1.5 & 2.1 & -0.3 \\ -0.1 & 0.1 & -0.5 & -2.1 & -0.6 & -0.0 & 0.3 & 0.8 \end{pmatrix} \rightarrow \begin{pmatrix} -26 & -2 & -4 & 1 & 2 & 0 & 0 & 0 \\ 0 & -1 & -2 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

При декодировании происходит обратный процесс — квантованные коэффициенты почленно умножаются на значения матрицы квантования. Но так как мы округляли значения, то не сможем точно восстановить исходные коэффициенты Фурье. Чем больше число квантования, тем больше погрешность. Таким образом, восстановленный коэффициент является лишь ближайшим кратным.

$$\begin{pmatrix} -416 & -22 & -40 & 16 & 48 & 0 & 0 & 0 \\ 0 & -12 & -28 & 0 & 0 & 0 & 0 & 0 \\ -28 & 0 & 32 & -24 & 0 & 0 & 0 & 0 \\ -28 & 0 & 22 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 55 & 47 & 56 & 74 & 70 & 55 & 63 & 89 \\ 56 & 51 & 68 & 98 & 101 & 78 & 68 & 78 \\ 60 & 55 & 77 & 120 & 133 & 104 & 73 & 64 \\ 70 & 57 & 72 & 116 & 136 & 109 & 73 & 58 \\ 81 & 57 & 59 & 93 & 111 & 93 & 71 & 64 \\ 88 & 61 & 53 & 73 & 83 & 72 & 68 & 78 \\ 89 & 67 & 60 & 72 & 71 & 59 & 68 & 91 \\ 87 & 72 & 70 & 79 & 71 & 56 & 69 & 98 \end{pmatrix}$$



Именно на этом шаге определяется степень сжатия, то есть насколько много нулей будет образовано после квантования. Если $F_q = 100\ \%$, то таблица квантования будет состоять только из единиц и значения в блоках округляются до целых.

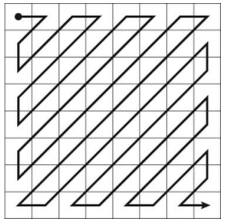
Алгоритм JPEG: 3 этап

Последний этап — это **энтропийное кодирование,** то есть кодирование последовательности значений с возможностью однозначного восстановления с целью уменьшения объёма данных (длины последовательности) с помощью усреднения вероятностей появления элементов в закодированной последовательности.

На этом этапе два основных шага: кодирование повторов (run-length encoding) и кодирование Хаффмана.

Кодирование повторов. Полученные коэффициенты из блока выписываются в строку с помощью обхода блока по “зиг-загу”. Следующая матрица задает последовательность обхода блока: от элемента 1 до 64.

$$\begin{pmatrix} 1 & 2 & 6 & 7 & 15 & 16 & 28 & 29 \\ 3 & 5 & 8 & 14 & 17 & 27 & 30 & 43 \\ 4 & 9 & 13 & 18 & 26 & 31 & 42 & 44 \\ 10 & 12 & 19 & 25 & 32 & 41 & 45 & 54 \\ 11 & 20 & 24 & 33 & 40 & 46 & 53 & 55 \\ 21 & 23 & 34 & 39 & 47 & 52 & 56 & 61 \\ 22 & 35 & 38 & 48 & 51 & 57 & 60 & 62 \\ 36 & 37 & 49 & 50 & 58 & 59 & 63 & 64 \end{pmatrix}$$



То есть, числа в этой матрице указывают порядок, в котором мы просматриваем каждый 8x8 блок.

После обхода по “зиг-загу”, получается вектор с 64 коэффициентами. Суть именно такого обхода в том, что мы просматриваем коэффициенты блоков 8x8 после DCT в порядке повышения пространственных частот. Первый элемент вектора (индекс 0) соответствует самой низкой частоте в изображении, её называют DC (direct current, фактически это усреднённое значение пикселей в блоке). Следующие элементы соответствуют более высоким частотам (последний элемент соответствует самой высокой частоте). Коэффициенты DCT, кроме первого, называют AC (alternating current).

Алгоритм JPEG: 3 этап

После обхода по “зиг-загу” можно ожидать, что полученный вектор с 64 коэффициентами имеет много нулей в “хвосте”.
Из этого вектора сформируем список блоков вида $[(a,\ b),\ c]$, где a, b, c значат следующее:

перед ненулевым отсчетом c находится a нулей, для кодирования числа c надо b бит. То есть

$[(a,\ b),\ c]$ = [(Количество нулей перед коэффициентом, Длина кода для коэффициента (в битах)), Коэффициент]

Таким образом, например, для матрицы

$$M=\begin{pmatrix} 31 & -15 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow (\ 31\ \ -15\ \ 0\ \ 0\ \ 0\ \ 0\ \ 6\ \ 0\ \ 0\ \ 0\ \ 13\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ -1\ \ 0\ \ ...\ )$$

кодировка такая $[(0, 5), 31], [(0, 4), -15], [(4, 3), 6], [(3, 4), 13], [(8, 1), -1], [(0, 0)]$, где $(0, 0)$ значит конец блока.

Коэффициенты и длина кода для коэффициента определяются с помощью следующей таблицы.

Величины	Категория	Биты для величины
0	0	-
-1,1	1	0,1
-3,-2,2,3	2	00,01,10,11
-7,..,-4, 4,..,7	3	000,001,010,011,100,101,110,111
-15,..,-8, 8,..,15	4	0000,..,0111,1000,..,1111
-31,..,-16, 16,..,31	5	00000,..,01111,10000,..,11111
-63,..,-32, 32,..,63	6	.
-127,..,-64, 64,..,127	7	.
-255,..,-128, 128,..,255	8	.
-511,..,-256, 256,..,511	9	.
-1023,..,-512, 512,..,1023	10	.
-2047,..,-1024, 1024,..,2047	11	.

-4095,..,-2048, 2048,..,4095	12	.
-8191,..,-4096, 4096,..,8191	13	.
-16383,..,-8192, 8192,..,16383	14	.
-32767,..,-16384, 16384,..,32767	15	.

В итоге : [(0, 5), 11111], [(0, 4), 0000], [(4, 3), 110], [(3, 4), 1101], [(8, 1), 1], [(0, 0)]

Однако, есть особенности.

1. Все первые коэффициенты в блоках кодируются отдельно. Это связано с тем, что DC коэффициенты могут иметь большое по модулю значение. При этом кодируются не сами значения, а разность DC коэффициентов между соседними блоками. Для их кодирования также используется специальная таблица. DC коэффициенту соответствует блок
[(длина кода в битах; битное кодированное коэффициента)]

2. Значения в скобках представляются одним байтом. Категория занимает 4 бита и число нулей также кодируется 4 битами. Поэтому если число нулей больше 15, то добавляется дополнительный блок [(15, 0)].

Алгоритм JPEG: 3 этап, код Хаффмана

Далее, результат предыдущего шага кодируется с помощью алгоритма Хаффмана для создания префиксного кода. Кодом Хаффмана кодируется байт (a, b), содержащий информацию о количестве нулей и длине кода.

Итак, каждый блок закодирован хранится в файле так:

[код Хаффмана для числа бит в DCdiff]
[DCdiff]
[код Хаффмана для (количество нулей перед AC1, число бит в AC1)]
[AC1]
…
[код Хаффмана для (количество нулей перед ACn, число бит в ACn)]
[ACn]

Где ACi — ненулевые AC коэффициенты.

Суть кода Хаффмана.

Пусть есть набор символов, который необходимо оптимально закодировать. Принцип кодирования символов основывается на информации о частоте повторения этих символов в кодируемой строке. Похожий принцип лежит в основе азбуки Морзе (точки-тире). Конечно же необходимо достичь взаимной-однозначности между исходной и закодированной строкой.

Алгоритм JPEG: 3 этап, код Хаффмана

Префиксный код в теории кодирования — код со словом переменной длины, имеющий такое свойство: если в код входит слово a, то для любой непустой строки b слова ab в коде не существует. Хотя префиксный код состоит из слов разной длины, эти слова можно записывать без разделительного символа.

Например, код, состоящий из слов 0, 10 и 11, является префиксным, и сообщение
01001101110
0 10 0
можно разбить на слова единственным образом:
0 10 0 11 0 11 10

Алгоритм Хаффмана позволяет построить таблицу для кодирования, кодирование с помощью которой почти достигает нижней теоретической границы энтропии.

Суть алгоритма Хаффмана: для строки из N символов считаются вероятности появления символов в строке. Далее:

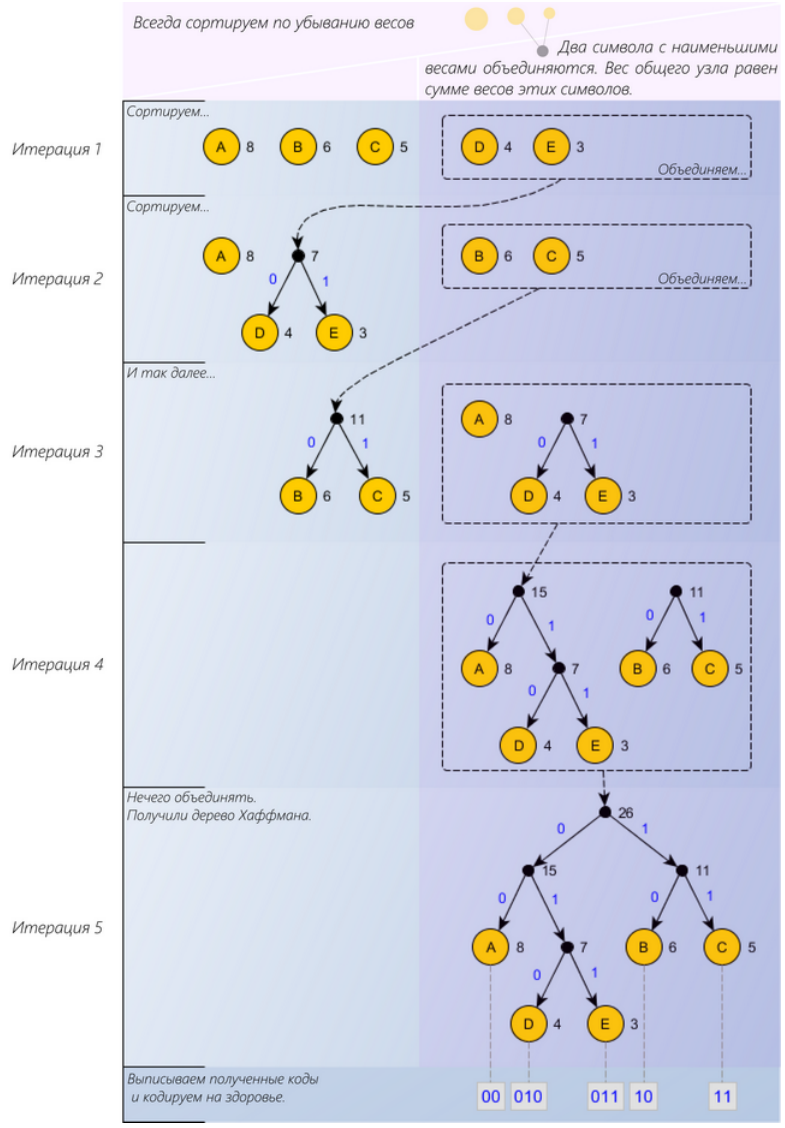
1 шаг. Все символы сортируются в порядке убывания вероятности их появления в строке.

2 шаг. Два последних символа объединяются в группу. Этой группе присваивается вероятность, равная сумме вероятностей этих символов. Далее эта группа участвует в алгоритме наравне с символами и другими группами.

Шаги повторяются, пока не останется только одна группа. В каждой группе одному символу (или подгруппе) присваивается бит 0, а другому бит 1.
Этот алгоритм называется кодированием Хаффмана.

Алгоритм JPEG: 3 этап, код Хаффмана

Пример кодирования Хаффмана.


Всегда сортируем по убыванию весов
Два символа с наименьшими весами объединяются. Вес общего узла равен сумме весов этих символов.
Итерация 1
Сортируем...
A 8
B 6
C 5
D 4
E 3
Объединяем...
Итерация 2
Сортируем...
A 8
7
0
1
D 4
E 3
B 6
C 5
Объединяем...
Итерация 3
И так далее...
11
0
1
B 6
C 5
A 8
7
D 4
E 3
Итерация 4
15
11
Итерация 5
Нечего объединять.
Получили дерево Хаффмана.
26
Выписываем полученные коды и кодируем на здоровье.
00
010
011
10
11

Алгоритм JPEG: 3 этап

Недостатки формата:

при сильных степенях сжатия дает знать о себе блочная структура данных, изображение «дробится на квадратики» (каждый размером 8 x8 пикселей). Этот эффект особенно заметен на областях с низкой пространственной частотой (плавные переходы изображения, например, чистое небо). В областях с высокой пространственной частотой (например, контрастные границы изображения), возникают характерные «артефакты» — иррегулярная структура пикселей искаженного цвета и/или яркости. Кроме того, из изображения пропадают мелкие цветные детали. Также, формат не поддерживает прозрачность.