ЛР6: Коды Хаффмана.

Цель: Использование кодов Хаффмана для кодирования с минимальной избыточностью.

Теоретическая часть

Один из наиболее известных методов сокращения кодовой избыточности был предложен Хаффманом [Huffman, 1952]. При независимом кодировании символов источника информации коды Хаффмана обеспечивают наименьшее число кодовых символов (битов) на символ источника. Коды Хаффмана используются в стандартах сжатия: JPEG; MPEG-1,2,4; H.264 и т. д.

Построение дерева Хаффмана

Первым шагом в алгоритме Хаффмана является построение серии (множества уровней) редуцированных источников путем упорядочивания вероятностей рассматриваемых символов и «склеивания» символов с наименьшими вероятностями в один символ, который будет замещать их в редуцированном источнике следующего уровня. Рассмотрим на примере алгоритм работы. Пусть имеется набор символов от a_1 до a_6 и известны вероятности их появления. Символы упорядочены сверху вниз в порядке убывания вероятностей.

Исходный источник		Редукция источника			
Символ	Вероятность	1	2	3	4
a_2	0,4	0,4	0,4	0,4	0,6
a_6	0,3	0,3	0,3	0,3	0,4
a_1	0,1	0,1	0,2	0,3	
a_4	0,1	0,1	0,1		
a_3	0,06	0,1			
a_5	0,04				

Для формирования первой редукции источника символы с наименьшими вероятностями — в данном случае 0,06 и 0,04 — объединяются в некоторый «составной символ» с суммарной вероятностью 0,1. Этот составной символ и связанная с ним вероятность помещаются в список символов редуцированного источника, который опять упорядочивается в порядке убывания значений полученных вероятностей. Этот процесс повторяется до тех пор, пока не

образуется редуцированный источник всего лишь с двумя оставшимися символами (самая правая колонка на рисунке).

Второй шаг в процедуре кодирования по Хаффману состоит в кодировании каждого из редуцированных источников, начиная с источника с наименьшим числом символов (т.е. правого) и возвращаясь обратно к исходному источнику. Для источника с двумя символами наименьшим двоичным кодом являются, конечно, символы 0 и 1. Как показано на рисунке, эти символы приписываются символам источника справа (выбор символов произволен — изменение 0 на 1 и наоборот даст абсолютно тот же результат). Поскольку символ текущего редуцированного источника, имеющий вероятность 0,6, был получен объединением двух символов предыдущего редуцированного источника, то кодовый бит 0, выбранный для данного варианта, приписывается каждому из двух соответствующих символов предыдущего источника, затем эти коды произвольным образом дополняются символами 0 и 1, для отличия их друг от друга. Эта операция затем повторяется для редуцированных источников всех остальных уровней, пока не будет достигнут уровень исходного источника.

Исходный источник			Редукция источника			
Символ	Вероятность	Код	1	2	3	4
a_2	0,4	1	0,4 1	0,4 1	0,4 1	0,6 0
a_6	0,3	00	0,3 00	0,3 00	0,3 00	0,4 1
a_1	0,1	011	0,1 011	0,2 010	0,3 01	
a_4	0,1	0100	0,1 0100	0,1 011		
a_3	0,06	01010	0,1 0101			
a_5	0,04	01011				

Кодирование и декодирование символов последовательности

Процедура Хаффмана строит оптимальный код для набора символов и их вероятностей при том ограничении, что символы кодируются по отдельности. После того, как код построен, процесс кодирования/декодирования без потерь осуществляется простым табличным преобразованием. Код Хаффмана является мгновенным однозначно декодируемым блоковым кодом. Он называется блоковым кодом, поскольку каждый символ источника отображается в фиксированную последовательность кодовых символов. Он является мгновенным, потому что каждое кодовое слово в строке кодовых символов может быть декодировано независимо от последующих символов. Он является однозначно декодируемым, т. к. любая строка из кодовых символов может быть декодирована единственным образом. Таким образом, любая строка кодированных по Хаффману символов может декодироваться анализом отдельных символов в строке слева направо.

Для примера рассмотрим закодированную строку 010100111100. Первым правильным кодовым словом является 01010, которое есть код для символа a3. Следующим правильным кодовым словом является 011, что соответствует символу a1. Продолжая эти действия, получим декодированное сообщение в виде a3a1a2a2a6.

Энтропия источника. Средняя длина кода

Рассмотрим источник статистически независимых случайных событий из дискретного набора случайных значений $\{a_1, a_2,..., a_J\}$ с ассоциированными вероятностями $\{P(a_1), P(a_2),..., P(a_J)\}$. Набор исходных символов {a1, a2,..., aJ} называется *алфавитом источника*. Среднее количество информации, приходящейся на один символ источника называется *энтропия*

$$H = -\sum_{j=1}^{J} P(a_j)\log_2(P(a_j)) \qquad (1.1)$$

Кодировать значения конечной последовательности, состоящей из набора символов $\{a_1, a_2,..., a_J\}$ с вероятностями $\{P(a_1), P(a_2),..., P(a_J)\}$ со средней кодовой длиной, меньшей, чем H бит/пиксель, невозможно.

Средняя длина кода конечной последовательности равна

$$L_{avg} = \sum_{j=1}^{J} P(a_j)\,N(a_j) \quad \text{(бит/символ)} \qquad (1.2)$$

где $N(a_j)$ — длина кода Хаффмана символа a_j (бит).

Средняя длина кода из примера равна Lavg = 0,4·1 + 0,3·2 + 0,1·3 + 0,1·4 +·0,06·5 + 0,04·5 = 2,2 бит/символ.
Энтропия источника равна H=2,14 бита/символ.

Символ	Вероятность	Код
a_2	0,4	1
a_6	0,3	00
a_1	0,1	011
a_4	0,1	0100
a_3	0,06	01010
a_5	0,04	01011

Коэффициент сжатия равен

$$C = \frac{M\,N\,bpp}{B_{huff}}\,100\%$$	(1.3)

где M, N — количество строк и столбцов в изображении; bpp — количество бит приходящееся на один пиксель (для цветных изображений — 24, для монохромных — 8); B_{huff} — минимальное количество бит необходимое для декодирования информации (с учетом словаря Хаффмана).

Ход работы

1. **Кодирование Хаффмана в MATLAB:**
   a) Загрузить изображение из папки *test_images* в соответствии с вариантом с помощью команды *imread*. Преобразовать в полутоновое изображение и обрезать до размеров 256x256.
   b) Построить и вывести гистограмму изображения.
   c) Рассчитать энтропию источника, в соответствии с уравнением (1.1).
   d) Выполнить кодирование изображения с помощью алгоритма Хаффмана (команды huffmandict, huffmanenco).
   e) Рассчитать среднюю длину кода, в соответствии с (1.2).
   f) Выполнить декодирование изображения (команда huffmandeco).
   g) Рассчитать коэффициент сжатия по формуле (1.3).
2. **Написать собственную реализацию функций huffmandict, huffmanenco, huffmandeco,**
   a) Выполнить кодирование изображения с помощью алгоритма Хаффмана.
   b) Рассчитать среднюю длину кода, в соответствии с (1.2).
   c) Выполнить декодирование изображения.
   d) Рассчитать коэффициент сжатия по формуле (1.3).
   e) Сравнить с реализацией в MATLAB.