Код Хэмминга. Пример работы алгоритма

Код Хэмминга — это алгоритм самоконтролирующегося и самокорректирующегося кода, который позволяет закодировать какое-либо информационное сообщение определённым образом и после передачи (например, по сети) определить появилась ли какая-то ошибка в этом сообщении (к примеру из-за помех) и, при возможности, восстановить это сообщение. В данном примере описан самый простой алгоритм Хемминга, который может исправлять лишь одну ошибку (существуют более совершенные модификации данного алгоритма, которые позволяют обнаруживать (и если возможно исправлять) большее количество ошибок).

Код Хэмминга состоит из двух частей. Первая часть кодирует исходное сообщение, вставляя в него в определённых местах контрольные биты (вычисленные особым образом). Вторая часть получает входящее сообщение и заново вычисляет контрольные биты (по тому же алгоритму, что и первая часть). Если все вновь вычисленные контрольные биты совпадают с полученными, то сообщение получено без ошибок. В противном случае, выводится сообщение об ошибке и при возможности ошибка исправляется.

Подготовка сообщения

Допустим, у нас есть сообщение «habr», которое необходимо передать без ошибок. Для этого сначала нужно наше сообщение закодировать при помощи Кода Хэмминга. Нам необходимо представить его в бинарном виде.

Символ	ACSII код	Бинарное представление
h	68	01000100
a	61	00111101
b	62	00111110
r	72	01001000

На этом этапе стоит определиться с, так называемой, длиной информационного слова, то есть длиной строки из нулей и единиц, которые мы будем кодировать. Допустим, у нас длина слова будет равна 16. Таким образом, нам необходимо разделить наше исходное сообщение («habr») на блоки по 16 бит, которые мы будем потом кодировать отдельно друг от друга. Так как один символ занимает в памяти 8 бит, то в одно кодируемое слово помещается ровно два ASCII символа. Итак, мы получили две бинарные строки по 16 бит:

h	a
01000100	00111101

и

b	r
00111110	01001000

После этого процесс кодирования распараллеливается, и две части сообщения («ha» и «br») кодируются независимо друг от друга. Рассмотрим, как это делается на примере первой части.

Прежде всего, необходимо вставить контрольные биты. Они вставляются в строго определённых местах — это позиции с номерами, равными степеням двойки. В нашем случае (при длине информационного слова в 16 бит) это будут позиции 1 (2^0), 2 (2^1), 4 (2^2), 8 (2^3), 16 (2^4). Соответственно, у нас получилось 5 контрольных бит (подчеркнуты):

Было:

h	a
01000100	00111101

Стало:

h	a
0̲0̲0̲1̲0̲0̲1̲0̲0̲	0̲0̲1̲0̲1̲1̲1̲0̲1̲

Таким образом, длина всего сообщения увеличилась на 5 бит. До вычисления самих контрольных бит, мы присвоили им значение «0».

Вычисление контрольных бит

Теперь необходимо вычислить значение каждого контрольного бита. Значение каждого контрольного бита зависит от значений информационных бит, но не от всех, а только от тех, которые этот контрольных бит контролирует. Для того чтобы понять, за какие биты отвечает каждых контрольный бит необходимо понять очень простую закономерность: контрольный бит с номером N контролирует все последующие N бит через каждые N бит, начиная с позиции N:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
X		X		X		X		X		X		X		X		X		X		X	1
	X	X			X	X			X	X			X	X			X	X			2
			X	X	X	X					X	X	X	X					X	X	4
							X	X	X	X	X	X	X	X							8
															X	X	X	X	X	X	16

Здесь знаком «X» обозначены те биты, которые контролирует контрольный бит, номер которого справа. То есть, к примеру, бит номер 12 контролируется битами с номерами 4 и 8. Ясно, что чтобы

узнать какими битами контролируется бит с номером N надо просто разложить N по степеням двойки.

Дальше вычисляем значение каждого контрольного бита: берём каждый контрольный бит и смотрим сколько среди контролируемых им битов единиц, получаем некоторое целое число и, если оно чётное, то ставим ноль, в противном случае ставим единицу. Можно конечно и наоборот, если число чётное, то ставим единицу, в противном случае, ставим 0. Главное, чтобы в «кодирующей» и «декодирующей» частях алгоритм был одинаков. (Мы будем применять первый вариант).

Высчитав контрольные биты для нашего информационного слова получаем следующее:

h	a
1001100<u>0</u>100	001<u>0</u>11101

и для второй части:

b	r
1<u>0</u>0<u>1</u>01<u>1</u>0<u>1</u>110	0101<u>0</u>1000

Декодирование и исправление ошибок

Теперь, допустим, мы получили закодированное первой частью алгоритма сообщение, но оно пришло к нам с ошибкой. К примеру, мы получили такое (11-ый бит передался неправильно):

h	a
1<u>0</u>0<u>1</u>1000<u>0</u>1<u>1</u>0	001<u>0</u>11101

Вся вторая часть алгоритма заключается в том, что необходимо заново вычислить все контрольные биты (так же как и в первой части) и сравнить их с контрольными битами, которые мы получили. Так, посчитав контрольные биты с неправильным 11-ым битом мы получим такую картину:

h	a
<u>0</u>1<u>0</u>1100<u>1</u>01<u>1</u>0	001<u>0</u>11101

Как мы видим, контрольные биты под номерами: 1, 2, 8 не совпадают с такими же контрольными битами, которые мы получили. Теперь просто сложив номера позиций неправильных контрольных бит (1 + 2 + 8 = 11) мы получаем позицию ошибочного бита. Теперь просто инвертировав его и отбросив контрольные биты, мы получим исходное сообщение в первозданном виде! Абсолютно аналогично поступаем со второй частью сообщения.

Примечания

1. В данном примере используется длина информационного сообщения 16 бит, но длину можно взять любую.
2. Для каждого числа проверочных символов используется специальная маркировка вида (k,i), где k — количество символов в сообщении, i — количество информационных символов в сообщении. Например, существуют коды (7, 4), (15, 11), (31, 26).
3. Число контрольных символов (r) можно вычислить по эмпирической формуле:

   $$r=[\log\{(i+1)+[\log(i+1)]\}], \text{ где}$$

   i - количество информационных символов в сообщении;
   [.] - означает округление до большего ближайшего целого значения
4. Каждый проверочный символ в коде Хэмминга представляет сумму по модулю 2 некоторой подпоследовательности данных. Стоит учитывать, что в данной простой версии алгоритма на одно информационное слово можно исправить только одну ошибку.
5. Код (7,4) является минимально возможным кодом с достаточно большой **избыточностью**. Эффективность кода (k/n) растет с увеличением длины кода
6. ***Избыточность кода*** — это количество проверочной информации в сообщении. Рассчитывается она по формуле:
   k/(i+k), где
   k — количество проверочных бит,
   i — количество информационных бит.
   Например, мы передаем 3 бита и к ним добавляем 1 проверочный бит — избыточность составит 1/(3+1) = 1/4 (25%).