**Циклические код (линейный систематический)**

берем полином с таблиц по необходимому колву битов (напр. 1001 = х3+1)

(умножение сообщения на порождающий полином)

делим на полином

избыточные биты – остаток от деления – синдром

составляем матрицу, помещаем туда порождающий полином в первый столбец (домножаем на оператор сдвига чтобы поместились n проверочных символов) и затем сдвигаем циклически (k+r):k вправо

приводим матрицу к каноническому виду (левая матрица должна стать диагональной) (справа получится матрица хэмминга)

можно либо сдвигать либо складывать

по полученный столбцам и полиному порождающему можно декодировать сообщение

ненулевой синдром всегда равен сумме по модулю два тех вектор-столбцов матрицы, номера которых соответствуют номерам ошибочных бит в слове Yn

**Перемежение/деперемежение данных**

Перемежение – изменение по установленному правилу естественного порядка элементов

Если биты каждого кодового слова Хn передаются не в обычной последовательности, а через интервалы, превышающие ожидаемое число рядом расположенных ошибочных бит (в промежутки между битами одного слова вставляются биты других кодовых слов), то при возникновении такого типа ошибки обратная перемежению операция – деперемежение – разнесет («размажет») группу ошибок по всей совокупности кодовых слов, составляющих данное сообщение. РАЗНОС С ГРУППЫ НА КАЖДЫЙ ИНТЕРВАЛ

Делим информационное сообщение на отдельные информационные слова

Записываем их в таблицу перемежения

Перемежаем лол

Предполагаем, что возникает подряд пакет ошибок. Записываем сообщение из строки в таблицу перемежения по столбцам, затем разбиваем на слова.

После деперемежации пакет ошибок преобразован в одиночные пакеты размером 1 бит

**Распаковка/сжатие методом Барроуза-Уилера**

Не классик, символьный алгоритм сжатия (преобразует в формат, годный для сжатия) типы: Символо-ориентированный, статистический, словарный, арифметический

Основные хар-ки сжатия: степень (отношение исходного к результирующему), скорость, сжатие

Сообщение записываем в матрицу, размером самого сообщения

Заполняем ее всеми циклическими сдвигами входного сообщения

Сортируем строки в алфавитном порядке

Выходная строка – последний столбец

**Лаб 9 Распаковка/сжатие на основе статистических методов (метод Шеннона-Фано)**

Статистические алгоритмы позволяют создавать более короткие коды для часто встречающихся и более длинные – для редко встречающихся символов алфавита или конкретного сообщения. В первом случае метод считается статическим статистическим, во втором – динамическим статистическим. Предусматривается создание кодовой таблицы. Два рассматриваемых метода отличаются в особенностях формирования таблицы. Сжатие идет без потерь.

Шеннон-Фано (неоптимальный):

Подсчитываем вероятностные параметры символов алфавита

Сортируем в порядке убывания вероятностей

Каждому символу ставим в соответствие бинарный код. Для этого:

Делим символы на две группы с примерно равной суммарной вероятностью (в зависимости от группы ставим старший символ 1 или 0)

10 11 101 110 111

Заменяем каждый символ сообщения на соответствующий ему бинарный код

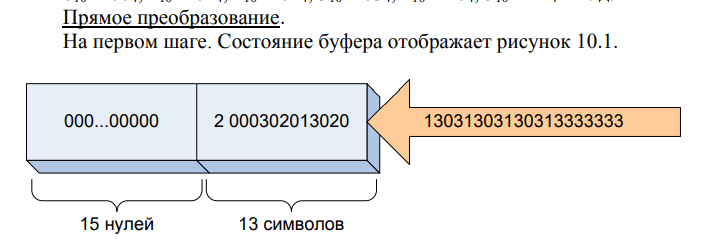
Декодирование:

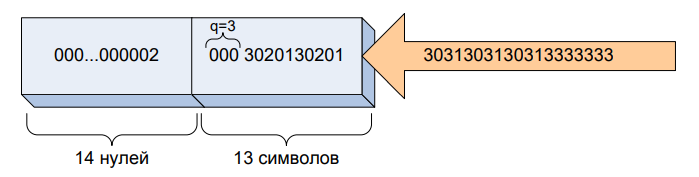
Анализируем первые два символа. Ищем их в бинарной таблице. Если не находим, то +1 бит. Когда нашли, заменяем на символ. Так до конца сообщения

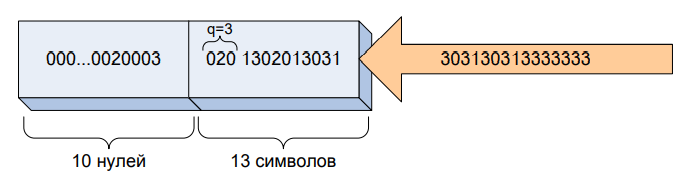
**Лаб 10 Распаковка/сжатие методом Лемпеля-Зива**

упаковщик постоянно хранит некоторое количество последних обработанных символов в буфере. По мере обработки входного потока вновь поступившие символы попадают в конец буфера, сдвигая предшествующие символы и вытесняя самые старые. Размеры этого буфера, называемого также скользящим словарем (sliding dictionary), варьируются в разных реализациях систем сжатия.



****

****

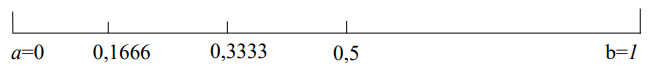
****

**Лаб 11 Распаковка/сжатие арифметическим методом**

Основная идея арифметического метода сжатия заключается в том, чтобы присваивать коды не отдельным символам, а их последовательностям. Таким образом, как и во всех энтропийных алгоритмах исходной является информация о частоте встречаемости каждого символа алфавита.

Рассчитываем вероятность появления каждого символа

Строим рабочий отрезок



Берется кодируемый символ, для него ищется соответствующий участок

Остальные символы (границы) распределяются на новом отрезку пропорционально

Проходим каждый символ. Итоговое число – нижняя граница последнего символа

Декодирование:

Определяем интервал, в который попадает наше число. Данный интервал становится новым рабочим отрезком, а соответствующий ему символ – первым.

Вычисляем код для нового интервала по формуле. Получаем код2. Смотрим, какому отрезку и символу он соответствует. И т.д.