Кодирование (сжатие) звука

Кодек

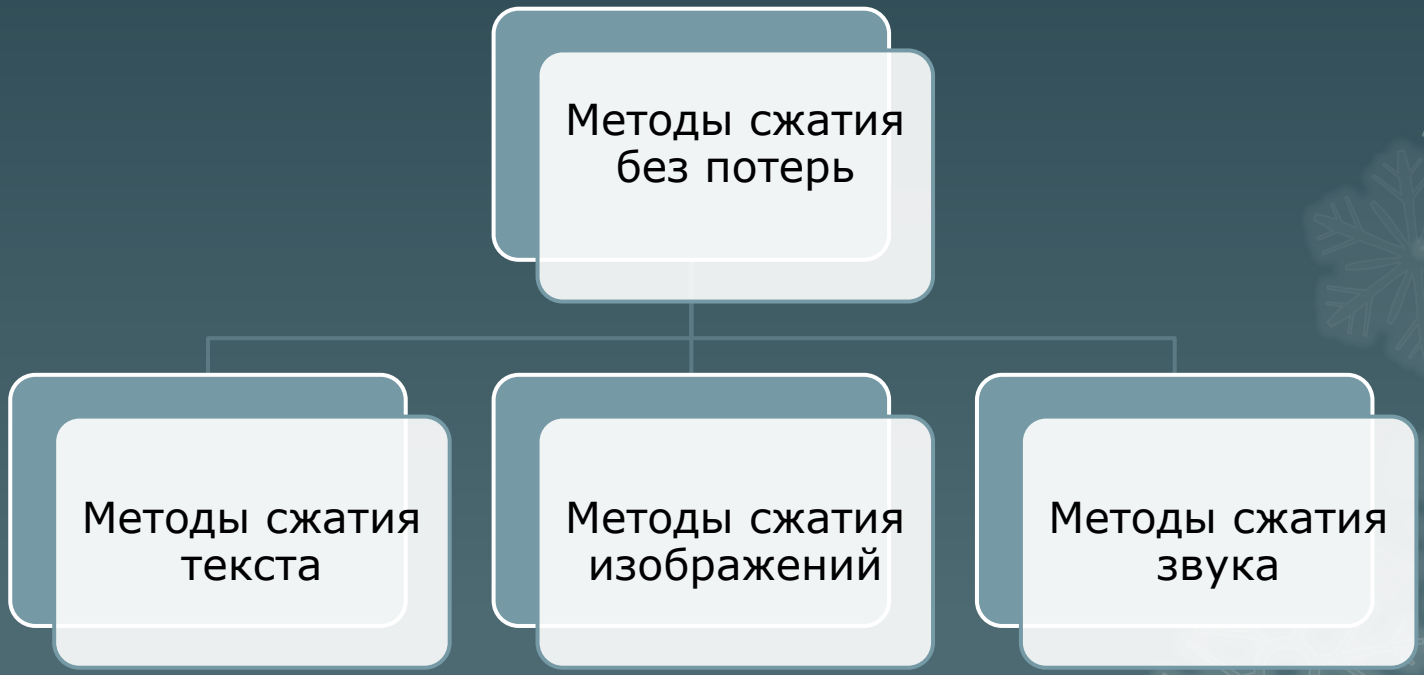
- **Кодек** (англ. codec, от coder/decoder — шифратор/дешифратор — кодировщик/декодировщик или compressor/decompressor) — устройство или программа, способная выполнять преобразование данных
- **Аудиокодек** (Audio codec) — компьютерная программа или аппаратное средство, предназначенное для кодирования или декодирования аудиоданных или сигнала.
- На программном уровне аудиокодек является программой который сжимает (производит компрессию) или разжимает (производит декомпрессию) цифровых звуковых данные в соответствии с файловым звуковым форматом или потоковым звуковым форматом.

Необходимость спец. алгоритмов для сжатия звука

- Алгоритм LZSS в среднем уменьшает объем звуковой информации на 15 процентов
- Кодирование Хаффмана уменьшает объем звуковой информации на 20-25 процентов
- ZIP – 10-20 %
- Небольшие показатели объясняются тем, что в звуковых файлах мало повторяющихся последовательностей кодов дискретизации.
- С другой стороны уровень звукового сигнала не перекрывает весь диапазон дискретизации.

Кодирование звука без потерь (lossless coding)

- Сжатие звука без потерь — совокупность преобразований, позволяющая эффективно сжимать звуковые данные с возможностью их полного восстановления.
- Классификация методов сжатия без потерь по типу данных:



Классификации методов сжатия без потерь по алгоритму действия

- **Словарные методы**: RLE, Deflate, LZ (LZ77/LZ78, LZSS, LZW, LZWL, LZO, LZMA, LZX, LZRW, LZJB, LZT);
- **Энтропийное сжатие** (Алгоритм Хаффмана, Адаптивный алгоритм Хаффмана, Арифметическое кодирование (Алгоритм Шеннона — Фано, Интервальное), Коды Голомба, Дельта, Универсальный код (Элиаса, Фибоначчи));
- **Статистические методы** включают алгоритмы для текста или бинарных данных: Преобразование Барроуза — Уилера (блочно-сортирующая пре-обработка), LZ77 и LZ78
- **Прочее**: CTW, BWT, MTF, PPM, DMC

- **Методы преобразования потока**;
- **Статические методы сжатия:**
  - Адаптивные (поточные) - вычисление вероятностей для новых данных происходит по данным, уже обработанным при кодировании
  - Блочные - статистика каждого блока данных высчитывается отдельно, и добавляется к самому сжатому блоку.
- **Методы преобразования блока.** Входящие данные разбиваются на блоки, которые затем трансформируются целиком. При этом некоторые методы, могут не приводить к существенному уменьшению объема данных. Однако после подобной обработки, структура данных значительно улучшается, и последующее сжатие другими алгоритмами проходит более успешно и быстро.

Методы преобразования потока (словарные методы)

- Предполагается описание новых поступающих несжатых данных через уже обработанные.
- При этом не вычисляется никаких вероятностей и кодирование символов осуществляется только на основе тех данных, которые уже были обработаны (например в LZ – методах (названных по имени Абрахама Лемпеля и Якоба Зива)).
- Идея алгоритмов:
- В случае нахождения второго и дальнейшие вхождения некой подстроки в уже известную кодировщику подстроку, найденная подстрока заменяется ссылками на ее первое вхождение.

Основная идея энтропийных* (статистических) методов сжатия

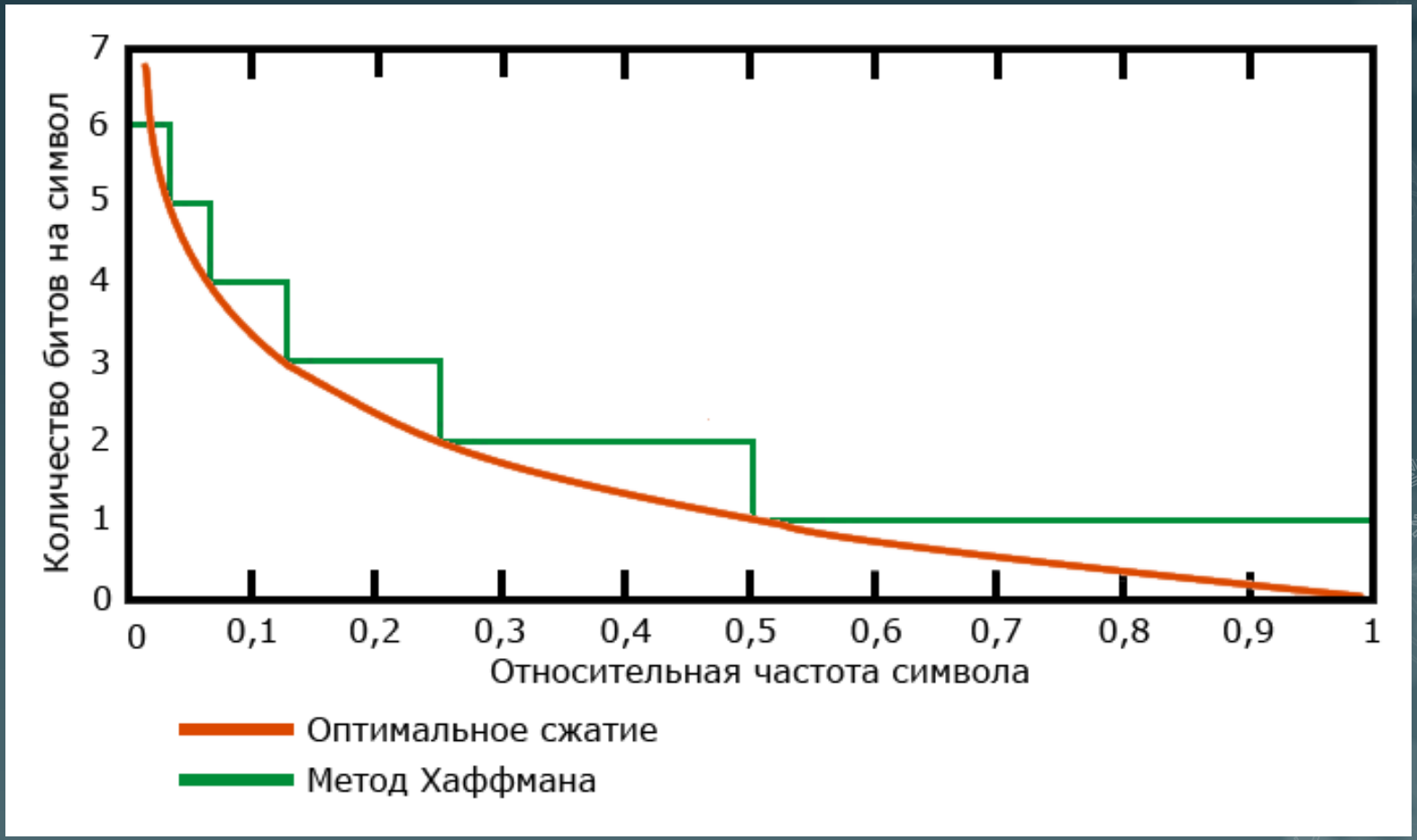
- Если представить, что наиболее часто встречающиеся элементы закодированы более короткими кодами, а реже встречающиеся – более длинными, то для хранения всех данных потребуется меньше места, чем если бы все элементы представлялись кодами одинаковой длины.
- Предполагается, что до кодирования отдельные элементы последовательности имеют различную вероятность появления. После кодирования в результирующей последовательности вероятности появления отдельных символов практически одинаковы (энтропия на символ максимальна).
- **Информационная энтропия — мера неопределённости источника сообщений, определяемая вероятностями появления тех или иных символов при их передаче.*
- **Информационная энтропия — мера неопределённости или непредсказуемости информации, неопределённость появления какого-либо символа первичного алфавита*

Предел сжатия без потерь

- **Существует предел сжатия без потерь**, зависящий от энтропии источника. Чем более предсказуемы получаемые данные, тем лучше их можно сжать. Случайная независимая равновероятная последовательность сжатию без потерь не поддаётся.
- По **теореме Шеннона** об источнике шифрования (или теорема бесшумного шифрования) оптимальная длинна кода для символа равна $-\log_b P$ где b число символов, использованных при генерации выходного кода (b=2 для двоичного кода), P — вероятность появления входного символа.
- То есть, если вероятность появления элемента s_i равна $p(s_i)$, то наиболее выгодно будет представить этот элемент — $\log_2 p(s_i)$ битами.
- Если при кодировании удается добиться того, что длина всех элементов будет приведена к $\log_2 p(s_i)$ битам, то и длина всей кодируемой последовательности будет минимальной для всех возможных методов кодирования. При этом, если распределение вероятностей всех элементов $F = \{p(s_i)\}$ неизменно, и вероятности элементов взаимно независимы, то средняя длина кодов может быть рассчитана как

$$H = -\sum_i p(s_i) \cdot \log_2 p(s_i).$$

Сравнение оптимального сжатия и метода Хаффмана

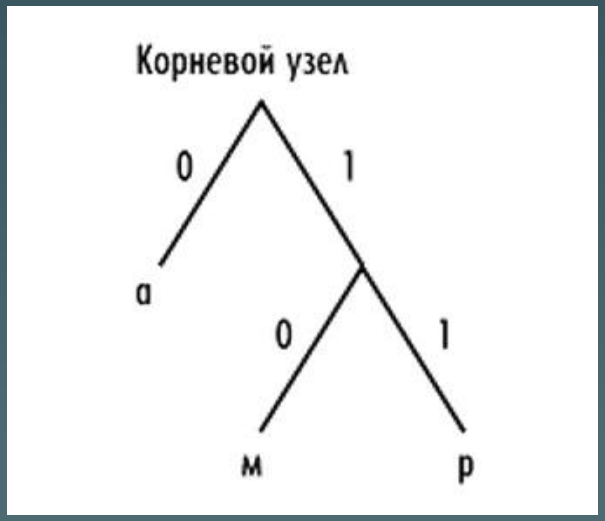


Префиксное кодирование

- Главная особенность префиксных кодов заключается в том, что в пределах каждой их системы более короткие по длине коды не совпадают с началом (префиксом) более длинных кодов.
- Поясним это свойство префиксных кодов на конкретном примере. Пусть имеется система из трех префиксных кодов: {0, 10, 11}. Более короткий код 0 не совпадает с началом более длинных кодов 10 и 11. Пусть код 0 задает символ «а», код 10 — символ «м», а код 11 — символ «р». Тогда слово «рама» кодируется последовательностью 110100.
- Попробуем раскодировать эту последовательность. Поскольку первый бит — это 1, то первый символ может быть только «м» или «р» и определяется значением второго бита. Поскольку второй бит — это 1, то первый символ — это «р». Третий бит — это 0, и он однозначно соответствует символу «а». Четвертый бит — это 1, то есть нужно смотреть на значение следующего бита, который равен 0, тогда третий символ — это «м». Последний бит — это 0, что однозначно соответствует символу «а».
- Таким образом, свойство префиксных кодов, заключающееся в том, что более короткие по длине коды не могут совпадать с началом более длинных кодов, позволяет однозначно декодировать закодированное префиксными кодами переменной длины информационное сообщение.

Кодовое дерево:

- Префиксные коды обычно получают построением **кодовых** (для двоичной системы) **деревьев**. Каждый внутренний узел такого бинарного дерева имеет два исходящих ребра, причем одному ребру соответствует двоичный символ «0», а другому — «1». Для определенности можно договориться, что левому ребру нужно ставить в соответствие символ «0», а правому — символ «1».
- Поскольку в дереве не может быть циклов, от корневого узла к листовому всегда можно проложить один-единственный маршрут. Если ребра дерева пронумерованы, то каждому такому маршруту соответствует некоторая уникальная двоичная последовательность. Множество всех таких последовательностей будет образовывать систему префиксных кодов.
- Для рассмотренного примера системы из трех префиксных кодов: {0, 10, 11}, которые задают символы «а», «м» и «р», кодовое дерево:

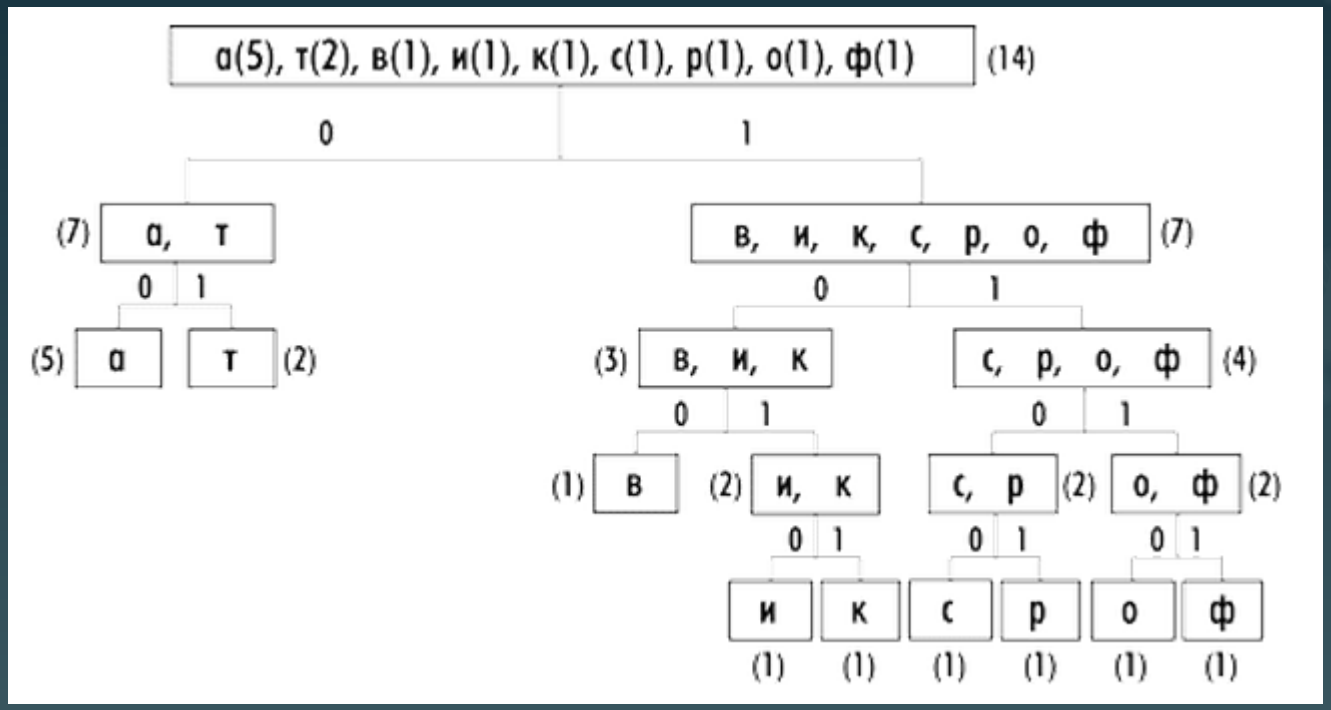


Алгоритм Шеннона—Фано

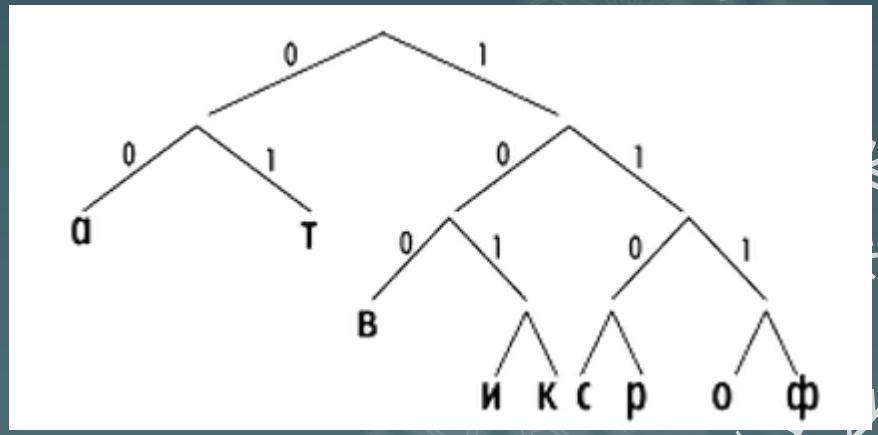
1. Символы первичного алфавита m_1 выписывают по убыванию вероятностей.
2. Символы полученного алфавита делят на две части, суммарные вероятности символов которых максимально близки друг другу.
3. В префиксном коде для первой части алфавита присваивается двоичная цифра «0», второй части — «1».
4. Полученные части рекурсивно делятся и их частям назначаются соответствующие двоичные цифры в префиксном коде.

- Рассмотрим построение кодового дерева для слова «авиакатастрофа». После сортировки по убыванию частоты появления символов получим: {а(5), т(2), в(1), и(1), к(1), с(1), р(1), о(1), ф(1)}
- Делим эту последовательность на две части так, чтобы в каждой из них сумма частот символов была примерно одинаковой:
- {а(5), т(2)} , {в(1), и(1), к(1), с(1), р(1), о(1), ф(1)}.
- Суммы частот повторяемости слева и справа символов равны 7.

Пример алгоритма Шенона-Фано



В примере получается следующая система префиксных кодов: «а» — 00, «т» — 01, «в» — 100, «и» — 1010, «к» — 1011, «с» — 1100, «р» — 1101, «о» — 1110, «ф» — 1111. Более короткие коды не являются началом более длинных кодов, то есть выполняется главное свойство префиксных кодов.

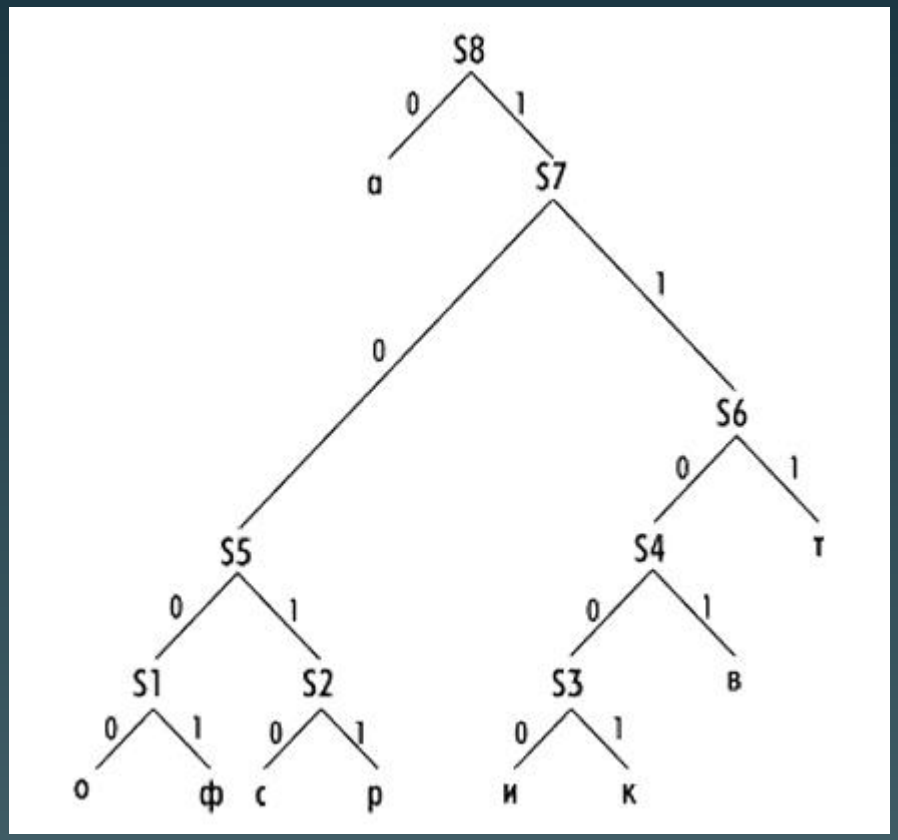


Кодирование по Хаффману

- Алгоритм Хаффмана на входе получает таблицу частот встречаемости символов в сообщении. Далее на основании этой таблицы строится дерево кодирования Хаффмана (H-дерево).

1. Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемое сообщение.
2. Выбираются два свободных узла дерева с наименьшими весами.
3. Создается их родитель с весом, равным их суммарному весу.
4. Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка.
5. Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой — бит 0.
6. Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться корнем дерева.

Пример



Пройдясь по ребрам кодового дерева сверху вниз, легко получить префиксные коды для всех символов: а-{0}, т{111}, в{1101}, и{11000}, к{11001}, с{1010}, р-{1011}, о{1000}, ф{1001}

S1(2)

{а(5), т(2), в(1), и(1), к(1), с(1), р(1), о(1), ф(1)}

S2(2)

{а(5), т(2), S1(2), в(1), и(1), к(1), с(1), р(1)}

S3(2)

{а(5), т(2), S1(2), S2(2), в(1), и(1), к(1)}

S4(3)

{а(5), т(2), S1(2), S2(2), S3(2), в(1)}

S5(4)

{а(5), S4(3), т(2), S1(2), S2(2)}

S6(5)

{а(5), S5(4), S4(3), т(2)}

S7(9)

{а(5), S6(5), S5(4)}

S8(14)

{S7(9), а(5)}

{S8(14)}

Lossless-алгоритмы для кодирования аудиоданных

- **Алгоритм Хаффмана**
- **DEFLATE**
- **арифметическое кодирование**
- **Apple Lossless** — *ALAC (Apple Lossless Audio Codec)*
- **Audio Lossless Coding** — *также известен как MPEG-4 ALS*
- **Free Lossless Audio Codec** — *FLAC*
- **Meridian Lossless Packing** — *MLP*
- **Monkey's Audio** — *APE*
- **OptimFROG**
- **RealPlayer** — *RealAudio Lossless*
- **Shorten** — *SHN*
- **TAK** — *(T)om's verlustfreier (A)udio (K)ompressor (нем.)*
- **TTA** — *True Audio Lossless*
- **WavPack** — *WavPack lossless*
- **WMA Lossless** — *Windows Media Lossless*
- **DTS** — *DTS Surround Sound*
- **TTE**
- **LA** *(LosslessAudio)*

Flac

- FLAC —свободный кодек, предназначенный для сжатия аудиоданных без потерь. Позволяет сжимать большинство музыкальных произведений с уровнем сжатия 30-50% (в сравнении Deflate использующий ZIP и gzip алгоритмы позволяет сжимать с уровнем 10-20%)
- Сильно стороной FLAC является быстрое кодирование/ декодирование по сравнению с другими Lossless кодеками. FLAC очень нетребовательно к ресурсам процессора.
- Основными частями потока являются:
  - Строка из четырёх байтов «fLaC»
  - Блок метаданных STREAMINFO
  - Другие необязательные блоки метаданных
  - Аудио фреймы
- Первые четыре байта идентифицируют поток FLAC. Следующие за ними метаданные содержат информацию о потоке, затем идут сжатые аудиоданные.
- Метаданные в блоке STREAMINFO содержат: частоту дискретизации, количество каналов, количество бит на семпл и количество семплов.
- FLAC делит входной поток на блоки и кодирует их независимо друг от друга. Блок упаковывается во фрейм и добавляется к потоку. Базовый кодер использует блоки постоянного размера для всего потока, однако формат предусматривает наличие блоков разной длины в потоке.
- Недостаток: менее эффективное сжатие, чем у некоторых других современных кодеров (APE (Monkey's Audio), LPAC, OptimFROG)

Аудио форматы без сжатия

- AIFF
- AU
- CDDA — формат, используемый в аудио-CD
- DSD — формат, используемый в SACD
- IFF-8SVX — Interchange File Format
- IFF-16SV
- RAW — сырые замеры без какого-либо заголовка или синхронизации
- WAV — Microsoft Wave (Waveform audio format). Разработан совместно с IBM