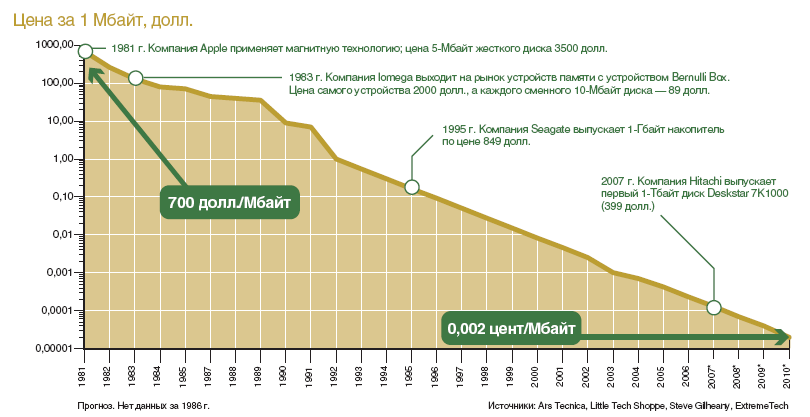
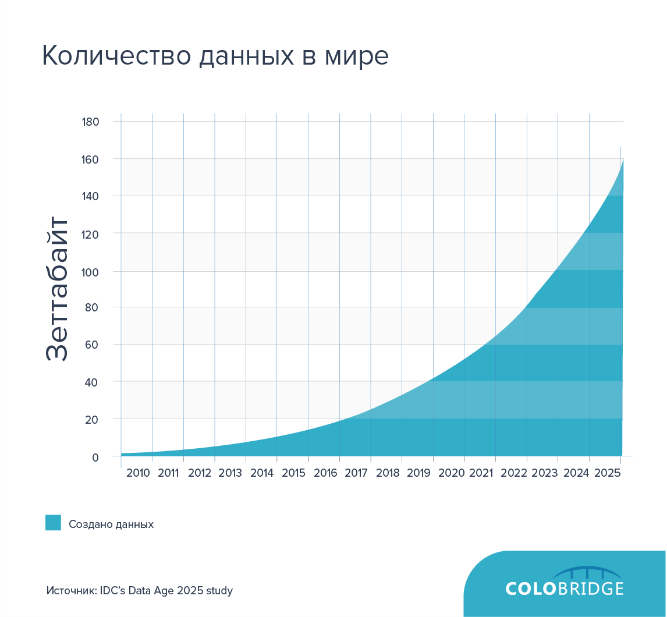
1. **Информационная катастрофа**

Для человечества главным хранилищем информации, на протяжении 1000 лет, являлись книги. Однако, за прошедшие 150 лет всё кардинально поменялось. Благодаря, открытию Уильямом Шокли транзистора в 1947 - ом, и концепции интегральной схемы от Джеффри Даммера в 1952 - ом. Для людей наступил новый этап в производстве, хранении и обработке данных.

И уже в 1996 году, хранить информацию в электронном виде стало дешевле, чем в бумажном. Сейчас же стоимость одного мегабайта составляет примерно 5,2 \* 10 -5$. Человечество научилось дешево сохранять информацию. Но столкнулось с новой проблемой.



В 2018 году человечество произвело 33 зеттабайта информации. С каждым годом этот показатель растёт. Например, в 2020 году общий объем данных составил 59 зеттабайт. И по прогнозам в 2025 году количество информации, произведенной за год, составит 175 зеттабайт.

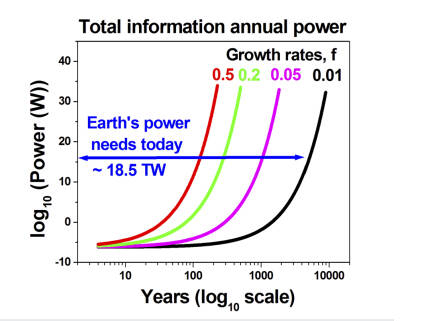


**Информационная мощность**  
  
Бит информации – это не просто абстрактная математическая сущность, но и физическая. В 1961 году Рольф Ландауэр впервые предложил связь между термодинамикой и информацией. Для удаления и создания бита информации требуется количество энергии равное по меньшей мере kb \* T \* ln (2), где kb – постоянная Больцмана, а T – температура, при которой хранится информация.

Общая планетарная потребность в энергии для поддержания производства цифровой информации

𝑄inf𝑜(𝑛𝑡ℎ)= 𝑁𝑏 ⋅ kb ⋅ 𝑇 ⋅ ln (2) ⋅ (𝑓+1)𝑛

Где 𝑓 – темп роста количества информации в год (в процентах). Общая потребность в электроэнергии сегодня для питания всех отраслей промышленности, транспорта и бытовых энергетических потребностей на Земле составляет около 18,5 ТВТ. При темпе роста в 20% через 285 лет создание цифрового контента потребуется затрат, эквивалентных всем сегодняшним планетарным потребностям в электроэнергии.



**Оценка количества бит информации**

Количество бит информации

𝑁𝑏𝑖𝑡𝑠(𝑛)= Nb/𝑓 ⋅ ((𝑓+1)𝑛+1−1)

При наших темпах создания цифрового контента через 340 лет у нас будет 1050 бит информации. Это количество битов представляет приблизительное количество атомов на Земле. Размер атома 10 -10 м, в то время как примерный размер бита информации (25\*10– 9) м2. Даже если предположить, что технологический прогресс уменьшит размер бита до размера близких к атому, этот объем цифровой информации будет занимать больше, чем размер планеты.

**Пути решения информационной катастрофы**

Одним из способов решения настигающей нас проблемы является изменение носителей информации. Кандидатом на эту роль могут стать системы хранения данных на основе ДНК. Позволяющие сохранять до 2,2 петабайт данных при весе в грамм. Так же ведутся разработки в области «Атомных» накопителей данных и квантовых хранилищах данных.

**Сжатие информации.**

Еще одним вариантом решения информационной катастрофы является сжатие уже имеющейся информации. Способы сжатия данных подразделяется на два типа:

* Неискажающие методы сжатия гарантируют, что закодированные данные будут полностью совпадать с исходными.
* Искажающие методы сжатия могут искажать исходные данные, к примеру удаляя несущественную часть данных, вследствие этого восстановить исходные данные невозможно.

Сжатие первого типа используется, когда нужно восстановить сжатые данные в первоначальном виде. Данный тип сжатия ничего не удаляет из исходных данных, само сжатие происходит за счёт более экономичного, представления данных.

Второй же тип применяют, для видео файлов, изображений, звука. За счет потерь можно достигнуть более высокой степени сжатия. В этом случае потери при сжатии означают несущественное искажение изображения (звука) которые не препятствуют нормальному восприятию, но при сличении оригинала и восстановленной после сжатия копии могут быть замечены.

**Критерии оценки методов сжатия**

Основными свойствами какого-либо алгоритма сжатия данных являются:

* Качество (коэффициент или степень сжатия) сжатия, т. е. отношение длины (в битах) сжатого представления данных к длине исходного представления;
* Скорость кодирования и декодирования, определяемые временем, затрачиваемым на кодирование и декодирование данных;
* Объем требуемой памяти.

В области сжатия данных, как это часто случается, действует закон рычага: алгоритмы, использующие больше ресурсов (времени и памяти), обычно достигают лучшего качества сжатия, и наоборот: менее ресурсоемкие алгоритмы по качеству сжатия, как правило, уступают более ресурсоемким.

Таким образом, построение оптимального с практической точки зрения алгоритма сжатия данных представляется достаточно нетривиальной задачей, так как необходимо добиться достаточно высокого качества сжатия (не обязательно оптимального с теоретической точки зрения) при небольшом объеме используемых ресурсов.

Понятно, что критерии оценки методов сжатия с практической точки зрения сильно зависят от предполагаемой области применения. Например, при использовании сжатия в системах реального времени необходимо обеспечить высокую скорость кодирования и декодирования; для встроенных систем критический параметр – объем требуемой памяти; для систем долговременного хранения данных – качество сжатия и/или скорость декодирования и т. д.

**Алгоритмы сжатия информации**

Без преувеличения можно сказать, что известны тысячи различных методов сжатия данных, однако многие из них заметно уступают другим по всем параметрам и поэтому не представляют интереса. Оставшиеся методы можно разбить на три больших класса.

**Алгоритмы статистического моделирования**

Наилучшие по качеству сжатия алгоритмы статистического моделирования источников, предсказывают вероятность появления следующего символа на основе анализа частоты появления различных последовательностей символов в ранее закодированной части сообщения.

Обладают очень низкой скоростью сжатия и требуют большого объема оперативной памяти. Скорость декодирования практически не отличается от скорости кодирования.

Несмотря на очень хорошие характеристики в смысле качества сжатия, использовать алгоритмы статистического моделирования на практике часто затруднительно или невозможно ввиду невысокой скорости сжатия.

Кроме того, многие предложенные способы реализации методов сжатия статистическим моделированием для получения оценок вероятностей появления символов используют команды умножения и/или деления, а иногда и вычисления с плавающей точкой. Так как такие реализации предъявляют очень жесткие требования к аппаратному обеспечению, область их применения ограничена.

**Алгоритмы словарного сжатия**

Алгоритмы словарного сжатия заменяют подстроки кодируемой последовательности символов ссылками в словарь на идентичные подстроки. Обычно для ускорения поиска совпадающих подстрок и ограничения объема требуемой памяти область поиска ограничивается определенным количеством последних символов закодированной части.

Алгоритмы семейства LZ77 в 1.3-1.7 раза уступают методам статистического моделирования по качеству сжатия, однако обладают очень высокой скоростью кодирования при сравнительно небольшом объеме требуемой памяти.

Огромное преимущество алгоритмов семейства LZ77 – чрезвычайно высокая скорость декодирования. Это позволяет применять их в тех случаях, когда Декодирование осуществляется гораздо чаще кодирования или скорость декодирования очень важна (например, при хранении данных на CD-ROM, в файловых системах со сжатием и т. д.).

Большая часть современных промышленных систем сжатия данных построено на основе различных вариантов алгоритма LZ77, в течение многих лет заслуженно считавшихся наилучшими по соотношению скорости и качества сжатия.

**Алгоритмы сжатия сортировкой блоков**

Алгоритмы сжатия сортировкой блоков семейства BWT/BS, разработанные в 1994г. Барроузом и Уилером, разбивают кодируемую последовательность на блоки символов, представляют (обратимым образом) символы каждого блока так, что появляется много повторений одного и того же символа, а затем сжимают преобразованные данные каким-либо достаточно простым способом.

По качеству сжатия они приближаются к методам статистического моделирования (уступая им в 1.2-1.3 раза), а по скорости – к алгоритмам семейства LZ77, при меньшем по сравнению с методами статистического моделирования объеме требуемой памяти; скорость декодирования также достаточно высока.

Ввиду своей новизны алгоритмы сжатия сортировкой блоков мало изучены, а известные реализации страдают серьезными недостатками, в частности, в наихудшем случае скорость сжатия уменьшается в десятки тысяч раз, что совершенно неприемлемо при создании надежных систем.

**Методы энтропийного кодирования**

Как правило, вышеперечисленные методы сжатия применяются не самостоятельно, а в сочетании с каким-либо методом энтропийного кодирования, заменяющего символы их кодовыми словами – строками нулей и единиц – так, что более часто встречающимся символам соответствуют более короткие слова.

Такие методы кодирования известны с конца 40-х гг. и хорошо изучены. Их можно разбить на два больших класса: префиксные (методы Хаффмана, Шеннона, Шеннона-Фано) и арифметические.

**Префиксные коды**

Префиксные коды называются так потому, что ни одно кодовое слово не является полным началом (т. е. префиксом) никакого другого слова, что гарантирует однозначность декодирования.

Известно много способов построения префиксных кодов: коды Шеннона и Шеннона-Фано почти идеальны, а код Хаффмана – оптимален среди префиксных кодов.

Так как длина каждого кодового слова выражается целым числом битов, то префиксные коды неэффективны на алфавитах малой мощности (2-8 символов) или при наличии символов с очень большой (более 30-50%) вероятностью появления и по качеству сжатия могут уступать арифметическим.

Применение блочных кодов, кодирующих не отдельные символы, а блоки из k символов, позволяет построение кодов, сколь угодно близких по качеству кодирования к арифметическим, однако из-за полиномиальной сложности блочного кодирования по размеру блока и ряда других причин блочное кодирование почти не применяется на практике.

Как правило, алгоритмы словарного сжатия и сжатия сортировкой блоков используют для кодирования выхода основного алгоритма сжатия коды Хаффмана.

**Арифметические коды**

Арифметические коды не ставят явного соответствия между символами и кодовыми словами, они основаны на других принципах.

Качество арифметического кодирования лучше, чем у посимвольного префиксного кодирования, и близко к теоретическому минимуму и при малой мощности алфавита, и при очень неравномерном распределении вероятностей появления символов.

Арифметические коды обычно применяются в сочетании с методами статистического моделирования для кодирования символов в соответствии с предсказанными вероятностями.

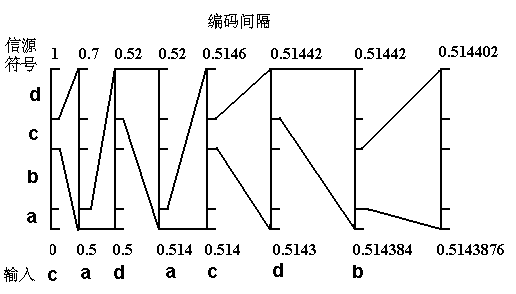
**История арифметическое кодирование**

Одним из алгоритмов энтропийного арифметического кода, является Арифметическое кодирование.

Базовые алгоритмы арифметического кодирования независимо разработали Йорма Дж. Риссанен из IBM Research и Ричард К. Паско. Оба алгоритма были опубликованы в мае 1976 года. Менее чем через год после публикации IBM подала заявку на патент США на работу Риссанена. Работа Паско не была запатентована.

**Идея арифметического кодирования**

Основной принцип арифметического кодирования состоит в том, чтобы представить закодированное сообщение как интервал между действительными числами 0 и 1. Чем длиннее сообщение, тем меньше интервал для кода для его представления и тем больше двоичных битов требуется для представления этого интервала.



1. <https://techinsider-ru.turbopages.org/techinsider.ru/s/editorial/701413-skolko-informacii-my-proizvodim-ezhegodno-i-gde-ona-hranitsya/>
2. <https://ieeexplore.ieee.org/document/5386860>
3. <https://www.comsol.ru/blogs/who-discovered-the-integrated-circuit/>
4. <https://theconversation.com/the-worlds-data-explained-how-much-were-producing-and-where-its-all-stored-159964>
5. <https://aip.scitation.org/doi/10.1063/5.0019941>
6. <https://habr.com/ru/company/selectel/blog/562962/>
7. <https://mf.grsu.by/UchProc/livak/po/comprsite/theory_classification_01.html>
8. <https://mai.ru/upload/iblock/f41/statisticheskiy-algoritm-szhatiya-informatsii.pdf>
9. <http://cotty.16x16.com/compress/peppm.htm>
10. <https://habr.com/ru/post/132683/>
11. <https://towardsdatascience.com/how-data-compression-works-exploring-lz77-3a2c2e06c097>
12. <https://neerc.ifmo.ru/wiki/index.php?title=Преобразование_Барроуза-Уилера>
13. <https://www.geeksforgeeks.org/burrows-wheeler-data-transform-algorithm/>
14. <https://habr.com/ru/company/playrix/blog/441814/>
15. <https://findpatent.ru/patent/237/2379832.html>
16. <https://studfile.net/preview/5240128/page:2/>
17. <https://intuit.ru/studies/courses/3481/723/lecture/14228?page=4>
18. <https://habr.com/ru/post/130531/>
19. <https://algolist.manual.ru/compress/standard/arithm.php>
20. <https://habr.com/ru/post/570694/>
21. <https://wikipredia.net/ru/Arithmetic_coding#History_and_patents>
22. https://russianblogs.com/article/8595747620/