|  |  |
| --- | --- |
|  | **МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**  **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕЛОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**  **КАЗАНСКИЙ ТЕХНОЛОГИЧЕСКИЙ КОЛЛЕДЖ** |

**Реферат по дисциплине**

**Основы теории информации**

**на тему «Архиваторы»**

**Выполнил работу**

**обучающийся группы 892**

**Маряшин Дмитрий Максимович**

**Проверил работу**

**преподаватель**

**Абзалимов Ришат Рафикович**

**Казань, 2020**

Оглавление

[**Введение** 3](#_Toc52989950)

[**ЧТО ТАКОЕ архиватор И КАК ОН РАБОТАет** 4](#_Toc52989951)

[**ДЛЯ ЧЕГО ОНИ НУЖНЫ** 4](#_Toc52989952)

[**ФУНКЦИЯ АРХИВАЦИИ В WINDOWS 10** 5](#_Toc52989953)

[**СТОРОННИЕ ПРОГРАММЫ АРХИВАТОРЫ** 5](#_Toc52989954)

[**Сжатие данных с потерями** 6](#_Toc52989955)

[**Алгоритм Шеннона — Фано** 7](#_Toc52989956)

[**Алгоритм Хаффмана** 9](#_Toc52989957)

[**Биграммная модель** 12](#_Toc52989958)

[**ExtractNow** 14](#_Toc52989959)

[**Литература (ссылки)** 15](#_Toc52989960)

**Введение**

У каждого веб мастера и SEO оптимизатора есть джентльменский набор программ, которые он использует ежедневно. Сегодня речь пойдет о такой необходимой программе как архиватор .

При чем не важно в какой компьютерной отрасли вы работаете, вам нужна эта программа. Даже если вы обычный секретарь в фирме которая занимается сборкой мебели — вам нужна эта программа.

Существуют много архиваторов таких как: ACB, ALZip, Archive Utility, MacBinary, PowerArchiver, Squeez StuffIt, WinAce, WinPRS, WinRAR, WinRK, WinZip и т.д., но сегодня мы затронем ExtractNow

**ЧТО ТАКОЕ архиватор И КАК ОН РАБОТАет**

Архив — это файл, содержащий в себе один или несколько других файлов вместе с их данными. То есть несколько отдельных файлов копируются в один. Для этого используются специальные программы.

Архиватор файлов — это компьютерная программа, которая объединяет несколько файлов в один архивный файл или серию архивных файлов для упрощения переноса, отправки или хранения. Файловые архиваторы могут использовать сжатие данных без потерь в свои специальные форматы, чтобы уменьшить размер архива. Также эти пакеты программ позволяют распаковывать или извлекать сжатые форматы файлов.

Самые простые программы просто берут список файлов и последовательно объединяют их содержимое в архивы. Архивные файлы должны хранить метаданные,  по крайней мере, название и размер исходных файлов, для корректной распаковки. Более продвинутые архиваторы хранят дополнительные метаданные, такие как оригинальные метки времени, атрибуты файлов или списки доступа.

Существуют разные форматы архивов и методы сжатия информации. Три наиболее распространенных типа архивных файлов: ZIP, RAR и 7Z. Они могут различаться размером конечного файла, скоростью сжатия информации, количеством файлов в архиве и т.д. Степень сжатия также зависит от архивируемых файлов. Например, текстовые файлы лучше всего поддаются сжатию, картинки сжимаются хуже, а файлы видео или музыки совсем не сжимаются, так как они уже максимально уплотнены.

**ДЛЯ ЧЕГО ОНИ НУЖНЫ**

Сжатие файлов может быть удобно, если у вас мало места на диске или вы хотите поделиться файлами в социальных сетях.

Архивы также полезны, когда вы хотите отправить несколько файлов кому-то по электронной почте. Вместо того, чтобы прикреплять несколько больших файлов, заархивируйте их в один файл. Архив займет меньше места, его легче прикрепить и отправить по электронной почте и удобнее скачать.

Форматы архивов ZIP или RAR широко используются разработчиками для упаковки своего программного обеспечения. Вам понадобятся архиваторы для извлечения файлов нужных программ.

Чтобы дополнительно защитить информацию в архивах, программы предлагают свои ключи шифрования и возможность поставить пароль на архив.

**ФУНКЦИЯ АРХИВАЦИИ В WINDOWS 10**

Наиболее популярным форматом для архивирования файлов является .zip, и Windows 10 может автоматически работать с архивами этого типа без необходимости устанавливать сторонние приложения.

Чтобы заархивировать несколько файлов и папок в ZIP-файл, откройте проводник и выполните следующие действия:

Выберите файлы и папки, которые вы хотите заархивировать.

Войдите в контекстное меню правой кнопкой мыши.

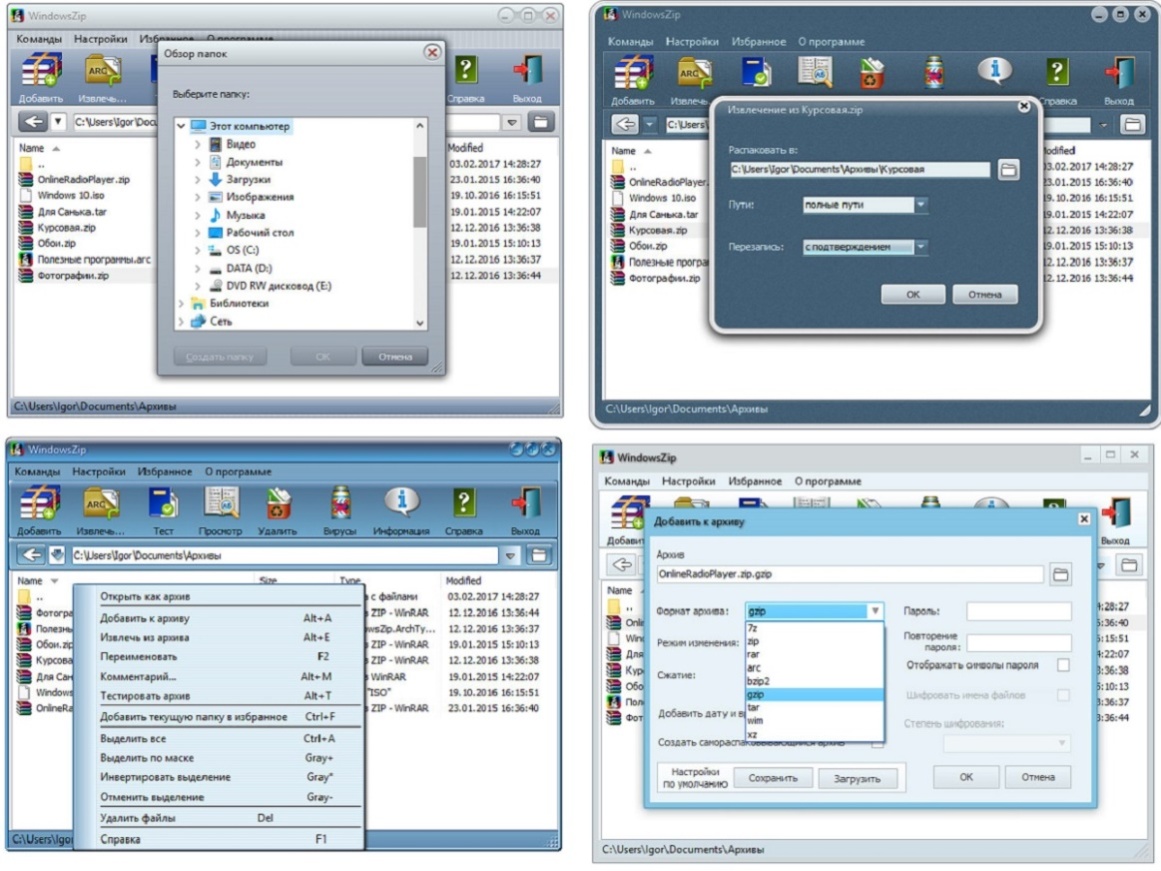
Функция архиватора будет в списке. Выберите «Addtoarchive…»

Архив автоматически создается в той же папке, что и выбранные вами файлы и папки. Вы можете редактировать название архива. Если вы хотите сэкономить место на жестком диске, рекомендуется удалить файлы и папки, которые вы поместили в архив, поскольку вы всегда можете извлечь их из архива.

**СТОРОННИЕ ПРОГРАММЫ АРХИВАТОРЫ**

Помимо встроенного инструмента для архивации в Windows 10 можно использовать различное программное обеспечение, которое предлагает более широкий спектр форматов архивации и другие дополнительные функции. Каждая программа поддерживает разные форматы файловых архивов, однако сейчас появляются все более универсальные инструменты.

Наиболее популярные бесплатные и условно бесплатные программы это 7-zip, WinRAR, WinZip, PeaZip и HamsterZipArchiver. Они все обладают похожим функционалом и помогут вам удобнее работать с файлами.



**Сжатие данных с потерями**

([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) lossycompression) — метод [сжатия](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (компрессии) данных, при использовании которого распакованные данные отличаются от исходных, но степень отличия не существенна с точки зрения их дальнейшего использования. Этот тип компрессии часто применяется для сжатия аудио- и видеоданных, статических изображений, в [Интернете](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82) (особенно в [потоковой передаче данных](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)) и цифровой [телефонии](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD%D0%B8%D1%8F). Альтернативой является [сжатие без потерь](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B1%D0%B5%D0%B7_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8C).

Существуют две основных схемы сжатия с потерями:

В трансформирующих [кодеках](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B5%D0%BA) фреймы изображений или звука обычно трансформируются в новое базисное пространство и производится [квантование](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D0%BE%D0%B2)). Трансформация может осуществляться либо для всего фрейма целиком (как, например, в схемах на основе [wavelet](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D1%81_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC_%D0%B2%D0%B5%D0%B9%D0%B2%D0%BB%D0%B5%D1%82)-преобразования), либо поблочно (характерный пример — [JPEG](https://ru.wikipedia.org/wiki/JPEG)). Результат затем сжимается [энтропийными методами](https://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

В предсказывающих кодеках предыдущие и/или последующие [отсчеты данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%86%D0%B8%D0%BC%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D0%BE%D0%B2)) используются для того, чтобы предсказать текущий отсчет изображения или звука. Ошибка между предсказанными данными и реальными вместе с добавочной информацией, необходимой для производства предсказания, затем [квантуется](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D0%BE%D0%B2)) и кодируется.

В некоторых системах эти две техники комбинируются путём использования трансформирующих кодеков для сжатия ошибочных сигналов, сгенерированных на стадии предсказания.

Сжатие с потерями против сжатия без потерь

Преимущество методов сжатия с потерями над методами сжатия без потерь состоит в том, что первые делают возможной большую степень сжатия, продолжая удовлетворять поставленным требованиям, а именно — искажения должны быть в допустимых пределах чувствительности человеческих органов, физических чувств.

Методы сжатия с потерями часто используются для сжатия аналоговых данных — чаще всего звука или изображений.

В таких случаях распакованный файл может очень сильно отличаться от оригинала на уровне сравнения «бит в бит», но практически неотличим для человека «на слух» и «на глаз» в большинстве применений.

Много методов фокусируются на физических особенностях органов чувств человека. [Психоакустическая модель](https://ru.wikipedia.org/wiki/%D0%9F%D1%81%D0%B8%D1%85%D0%BE%D0%B0%D0%BA%D1%83%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) определяет то, как сильно звук может быть сжат без ухудшения воспринимаемого человеком качества звука. Недостатки, причинённые сжатием с потерями, которые заметны для человеческого уха или глаза, известны как [артефакты сжатия](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%82%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D1%8B_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D1%8F).

Фотографии, записанные в формате JPEG, могут быть приняты судом в качестве доказательств несмотря на то, что изображение сжато с потерями.

Недостатки

При использовании сжатия с потерями необходимо учитывать, что повторное сжатие обычно приводит к деградации качества. Однако, если повторное сжатие выполняется без каких-либо изменений сжимаемых данных, качество не меняется.

**Алгоритм Шеннона — Фано**

один из первых алгоритмов сжатия, который впервые сформулировали американские учёные [Клод Шеннон](https://ru.wikipedia.org/wiki/%D0%A8%D0%B5%D0%BD%D0%BD%D0%BE%D0%BD,_%D0%9A%D0%BB%D0%BE%D0%B4_%D0%AD%D0%BB%D0%B2%D1%83%D0%B4) и [Роберт Фано](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BD%D0%BE,_%D0%A0%D0%BE%D0%B1%D0%B5%D1%80%D1%82). Данный метод сжатия имеет большое сходство с [алгоритмом Хаффмана](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0), который появился на несколько лет позже и является логическим продолжением [алгоритма Шеннона](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A8%D0%B5%D0%BD%D0%BD%D0%BE%D0%BD%D0%B0). Алгоритм использует коды переменной длины: часто встречающийся символ кодируется кодом меньшей длины, редко встречающийся — кодом большей длины. Коды Шеннона — Фано — беспрефиксные, то есть никакое кодовое слово не является [префиксом](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B0#%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5) любого другого. Это свойство позволяет однозначно декодировать любую последовательность кодовых слов.

Основные сведения

*Кодирование Шеннона — Фано* ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Shannon–Fanocoding*) — алгоритм префиксного неоднородного кодирования. Относится к [вероятностным](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) методам сжатия (точнее, методам [контекстного моделирования](https://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) нулевого порядка). Подобно [алгоритму Хаффмана](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0), алгоритм Шеннона — Фано использует избыточность сообщения, заключённую в неоднородном распределении частот символов его ([первичного](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B2%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82)) алфавита, то есть заменяет коды более частых символов короткими двоичными последовательностями, а коды более редких символов — более длинными двоичными последовательностями.

Алгоритм был независимо друг от друга разработан Шенноном (публикация «Математическая теория связи», 1948 год) и, позже, Фано (опубликовано как технический отчёт).

Основные этапы

Символы первичного алфавита m1 выписывают по убыванию вероятностей.

Символы полученного алфавита делят на две части, суммарные вероятности символов которых максимально близки друг другу.

В префиксном коде для первой части алфавита присваивается [двоичная](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) цифра «0», второй части — «1».

Полученные части рекурсивно делятся и их частям назначаются соответствующие двоичные цифры в префиксном коде.

Когда размер подалфавита становится равен нулю или единице, то дальнейшего удлинения префиксного кода для соответствующих ему символов первичного алфавита не происходит, таким образом, алгоритм присваивает различным символам префиксные коды разной длины. На шаге деления алфавита существует неоднозначность, так как разность суммарных вероятностей {\displaystyle p\_{0}-p\_{1}} может быть одинакова для двух вариантов разделения (учитывая, что все символы первичного алфавита имеют вероятность больше нуля).

Алгоритм вычисления кодов Шеннона — Фано

Код Шеннона — Фано строится с помощью дерева. Построение этого дерева начинается от корня. Всё множество кодируемых элементов соответствует корню дерева (вершине первого уровня). Оно разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Эти подмножества соответствуют двум вершинам второго уровня, которые соединяются с корнем. Далее каждое из этих подмножеств разбивается на два подмножества с примерно одинаковыми суммарными вероятностями. Им соответствуют вершины третьего уровня. Если подмножество содержит единственный элемент, то ему соответствует концевая вершина кодового дерева; такое подмножество разбиению не подлежит. Подобным образом поступаем до тех пор, пока не получим все концевые вершины. Ветви кодового дерева размечаем символами 1 и 0, как в случае кода Хаффмана.

При построении кода Шеннона — Фано разбиение множества элементов может быть произведено, вообще говоря, несколькими способами. Выбор разбиения на уровне n может ухудшить варианты разбиения на следующем уровне (n + 1) и привести к неоптимальности кода в целом. Другими словами, оптимальное поведение на каждом шаге пути ещё не гарантирует оптимальности всей совокупности действий. Поэтому код Шеннона — Фано не является оптимальным в общем смысле, хотя и дает оптимальные результаты при некоторых распределениях вероятностей. Для одного и того же распределения вероятностей можно построить, вообще говоря, несколько кодов Шеннона — Фано, и все они могут дать различные результаты. Если построить все возможные коды Шеннона — Фано для данного распределения вероятностей, то среди них будут находиться и все коды Хаффмана, то есть оптимальные коды.

**Алгоритм Хаффмана**

[жадный алгоритм](https://ru.wikipedia.org/wiki/%D0%96%D0%B0%D0%B4%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) оптимального [префиксного](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) [кодирования](https://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) алфавита с минимальной [избыточностью](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B1%D1%8B%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C). Был разработан в [1952 году](https://ru.wikipedia.org/wiki/1952_%D0%B3%D0%BE%D0%B4) аспирантом [Массачусетского технологического института](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B0%D1%87%D1%83%D1%81%D0%B5%D1%82%D1%81%D0%BA%D0%B8%D0%B9_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82) [Дэвидом Хаффманом](https://ru.wikipedia.org/wiki/%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD,_%D0%94%D1%8D%D0%B2%D0%B8%D0%B4) при написании им курсовой работы. В настоящее время используется во многих программах сжатия данных.

В отличие от [алгоритма Шеннона — Фано](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A8%D0%B5%D0%BD%D0%BD%D0%BE%D0%BD%D0%B0_%E2%80%94_%D0%A4%D0%B0%D0%BD%D0%BE), алгоритм Хаффмана остаётся всегда оптимальным и для [вторичных алфавитов](https://ru.wikipedia.org/wiki/%D0%92%D1%82%D0%BE%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82) m2 с более чем двумя символами.

Этот метод кодирования состоит из двух основных этапов:

Построение оптимального кодового дерева.

Построение отображения код-символ на основе построенного дерева.

Кодирование Хаффмана

Один из первых алгоритмов эффективного кодирования информации был предложен Д. А. Хаффманом в 1952 году. Идея алгоритма состоит в следующем: зная вероятности появления символов в сообщении, можно описать процедуру построения кодов переменной длины, состоящих из целого количества битов. Символам с большей вероятностью ставятся в соответствие более короткие коды. Коды Хаффмана обладают свойством [префиксности](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) (то есть ни одно кодовое слово не является префиксом другого), что позволяет однозначно их декодировать.

Классический алгоритм Хаффмана на входе получает таблицу частот встречаемости символов в сообщении. Далее на основании этой таблицы строится дерево кодирования Хаффмана (Н-дерево).[[1]](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0#cite_note-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.

Выбираются два свободных узла дерева с наименьшими весами.

Создается их родитель с весом, равным их суммарному весу.

Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка.

Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой — бит 0. Битовые значения ветвей, исходящих от корня, не зависят от весов потомков.

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.

Этот процесс можно представить как построение [дерева](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)), корень которого — символ с суммой вероятностей объединенных символов, получившийся при объединении символов из последнего шага, его n0 потомков — символы из предыдущего шага и т. д.

Чтобы определить код для каждого из символов, входящих в сообщение, мы должны пройти путь от листа дерева, соответствующего текущему символу, до его корня, накапливая биты при перемещении по ветвям дерева (первая ветвь в пути соответствует младшему биту). Полученная таким образом последовательность битов является кодом данного символа, записанным в обратном порядке.Поскольку ни один из полученных кодов не является префиксом другого, они могут быть однозначно декодированы при чтении их из потока. Кроме того, наиболее частый символ сообщения А закодирован наименьшим количеством бит, а наиболее редкий символ Д — наибольшим.

При этом общая длина сообщения, состоящего из приведённых в таблице символов, составит 87 бит (в среднем 2,2308 бита на символ). При использовании равномерного кодирования общая длина сообщения составила бы 117 бит (ровно 3 бита на символ). Заметим, что [энтропия](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%8D%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D1%8F) источника, независимым образом порождающего символы с указанными частотами, составляет ~2,1858 бита на символ, то есть [избыточность](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B1%D1%8B%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8) построенного для такого источника кода Хаффмана, понимаемая как отличие среднего числа бит на символ от энтропии, составляет менее 0,05 бит на символ.

Классический алгоритм Хаффмана имеет ряд существенных недостатков. Во-первых, для восстановления содержимого сжатого сообщения декодер должен знать таблицу частот, которой пользовался кодер. Следовательно, длина сжатого сообщения увеличивается на длину таблицы частот, которая должна посылаться впереди данных, что может свести на нет все усилия по сжатию сообщения. Кроме того, необходимость наличия полной частотной статистики перед началом собственно кодирования требует двух проходов по сообщению: одного для построения модели сообщения (таблицы частот и Н-дерева), другого для собственно кодирования. Во-вторых, избыточность кодирования обращается в ноль лишь в тех случаях, когда вероятности кодируемых символов являются обратными степенями числа 2. В-третьих, для источника с энтропией, не превышающей 1 (например, для двоичного источника), непосредственное применение кода Хаффмана бессмысленно.

Адаптивное сжатие

Адаптивное сжатие позволяет не передавать модель сообщения вместе с ним самим и ограничиться одним проходом по сообщению как при кодировании, так и при декодировании.

В создании алгоритма адаптивного кодирования Хаффмана наибольшие сложности возникают при разработке процедуры обновления модели очередным символом. Теоретически можно было бы просто вставить внутрь этой процедуры полное построение дерева кодирования Хаффмана, однако, такой алгоритм сжатия имел бы неприемлемо низкое быстродействие, так как построение Н-дерева — это слишком большая работа, и производить её при обработке каждого символа неразумно. К счастью, существует способ модифицировать уже существующее Н-дерево так, чтобы отобразить обработку нового символа. Наиболее известными алгоритмами перестроения являются алгоритм Фоллера-Галлагера-Кнута (FGK) и алгоритм Виттера.

Все алгоритмы перестроения дерева при считывании очередного символа, включают в себя две операции:

Первая — увеличение веса узлов дерева. Вначале увеличиваем вес листа, соответствующего считанному символу, на единицу. Затем увеличиваем вес родителя, чтобы привести его в соответствие с новыми значениями веса потомков. Этот процесс продолжается до тех пор, пока мы не доберемся до корня дерева. Среднее число операций увеличения веса равно среднему количеству битов, необходимых для того, чтобы закодировать символ.

Вторая операция — перестановка узлов дерева — требуется тогда, когда увеличение веса узла приводит к нарушению свойства упорядоченности, то есть тогда, когда увеличенный вес узла стал больше, чем вес следующего по порядку узла. Если и дальше продолжать обрабатывать увеличение веса, двигаясь к корню дерева, то дерево перестанет быть деревом Хаффмана.

Чтобы сохранить упорядоченность дерева кодирования, алгоритм работает следующим образом. Пусть новый увеличенный вес узла равен W+1. Тогда начинаем двигаться по списку в сторону увеличения веса, пока не найдем последний узел с весом W. Переставим текущий и найденный узлы между собой в списке, восстанавливая таким образом порядок в дереве (при этом родители каждого из узлов тоже изменятся). На этом операция перестановки заканчивается.

После перестановки операция увеличения веса узлов продолжается дальше. Следующий узел, вес которого будет увеличен алгоритмом, — это новый родитель узла, увеличение веса которого вызвало перестановку.

**Биграммная модель**

Существует разновидность алгоритма Хаффмана, использующая контекст. В данном случае размер контекста равен единице (биграммный — два символа, триграммный — три и так далее). Это метод построения префиксного кода для моделей высших порядков, уже не источника без памяти. Он использует результат (предыдущей операции) операции над предыдущей буквой совместно с текущей буквой. Строится на основе [цепи Маркова](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BF%D1%8C_%D0%9C%D0%B0%D1%80%D0%BA%D0%BE%D0%B2%D0%B0) с глубиной зависимости {\displaystyle r=1}

**Алгоритм**

Строится таблица в виде квадрата — распределение вероятностей на биграммах. Сразу вычисляется стартовая схема, с помощью которой будет кодироваться только первая буква. Строками в таблице, например, являются предыдущие буквы, а столбцами текущие.

Вычисляются вероятности для кодовых деревьев для контекстов.

По контекстам длины {\displaystyle r=1} строятся остальные кодовые деревья, с помощью которых будут кодироваться все остальные символы (кроме первого).

Выполняется кодирование, первый символ кодируется согласно стартовой схеме, все последующие — исходя из кодовых деревьев для контекстов (предыдущего символа).

Декодирование выполняется аналогично: из стартовой кодовой схемы получаем первый контекст, а затем переходим к соответствующему кодовому дереву. Более того, декодеру необходима таблица распределения вероятностей.

Переполнение

В процессе работы алгоритма сжатия вес узлов в дереве кодирования Хаффмана неуклонно растет. Первая проблема возникает тогда, когда вес корня дерева начинает превосходить вместимость ячейки, в которой он хранится. Как правило, это 16-битовое значение и, следовательно, не может быть больше, чем 65535. Вторая проблема, заслуживающая ещё большего внимания, может возникнуть значительно раньше, когда размер самого длинного кода Хаффмана превосходит вместимость ячейки, которая используется для того, чтобы передать его в выходной поток. Декодеру все равно, какой длины код он декодирует, поскольку он движется сверху вниз по дереву кодирования, выбирая из входного потока по одному биту. Кодер же должен начинать от листа дерева и двигаться вверх к корню, собирая биты, которые нужно передать. Обычно это происходит с переменной типа «целое», и, когда длина кода Хаффмана превосходит размер типа «целое» в битах, наступает переполнение.

Можно доказать, что максимальную длину код Хаффмана для сообщений с одним и тем же входным алфавитом будет иметь, если частоты символов образует [последовательность Фибоначчи](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%A4%D0%B8%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D0%B8). Сообщение с частотами символов, равными числам Фибоначчи до Fib (18), — это отличный способ протестировать работу программы сжатия по Хаффману.

Масштабирование весов узлов дерева Хаффмана

Принимая во внимание сказанное выше, алгоритм обновления дерева Хаффмана должен быть изменен следующим образом: при увеличении веса нужно проверять его на достижение допустимого максимума. Если мы достигли максимума, то необходимо «масштабировать» вес, обычно разделив вес листьев на целое число, например, 2, а потом пересчитав вес всех остальных узлов.

Однако при делении веса пополам возникает проблема, связанная с тем, что после выполнения этой операции дерево может изменить свою форму. Объясняется это тем, что при делении целых чисел отбрасывается дробная часть.

Правильно организованное дерево Хаффмана после масштабирования может иметь форму, значительно отличающуюся от исходной. Это происходит потому, что масштабирование приводит к потере точности статистики. Но со сбором новой статистики последствия этих «ошибок» практически сходят на нет. Масштабирование веса — довольно дорогостоящая операция, так как она приводит к необходимости заново строить все дерево кодирования. Но, так как необходимость в ней возникает относительно редко, то с этим можно смириться.

Выигрыш от масштабирования

Масштабирование веса узлов дерева через определенные интервалы дает неожиданный результат. Несмотря на то, что при масштабировании происходит потеря точности статистики, тесты показывают, что оно приводит к лучшим показателям сжатия, чем если бы масштабирование откладывалось. Это можно объяснить тем, что текущие символы сжимаемого потока больше «похожи» на своих близких предшественников, чем на тех, которые встречались намного раньше. Масштабирование приводит к уменьшению влияния «давних» символов на статистику и к увеличению влияния на неё «недавних» символов. Это очень сложно измерить количественно, но, в принципе, масштабирование оказывает положительное влияние на степень сжатия информации. Эксперименты с масштабированием в различных точках процесса сжатия показывают, что степень сжатия сильно зависит от момента масштабирования веса, но не существует правила выбора оптимального момента масштабирования для программы, ориентированной на сжатие любых типов информации.

**ExtractNow**

бесплатный архиватор для Windows, поддерживающий только распаковку. Создавать архивы программа "не умеет".

У большинства пользователей домашних ПК работа с архивами сводится только к их распаковке. Обратный же процесс, как показывает практика, рядовыми гражданами осуществляется не так уж и часто. Для них **возможностей ExtractNow** будет более чем достаточно:

- распаковка значительного количества архивов разных типов за один раз;

- работа методом “draganddrop” - для распаковки достаточно перетащить архив мышкой в окно программы и нажать кнопку “extract”;

- сохранение истории распакованных файлов;

- поддержка практически всех видов архивов: rar, 7z, cab, zip (+jar, xpi), bzip2 (+bz2, tbz2, tbz), arj, z, lzh (+lha), nsis, lzma, lzma86, xz (+txz), ppmd, udf, xar, mub, hfs, dmg, compound (+msi, doc, xls, ppt), fat (+fat, img), mbr, vhd, pe, wim (+swm), iso, chm (+chm, chi, chq, chw, hxs, hxi, hxr, hxq, hxw, lit), split (+001), rpm, deb, cpio, tar, gzip (+gz, gzip, tgz, tpz), mslz, flv, swf, ntfs (+ntfs, img), elf, apm, macho, ace, sit;

- возможность открыть папку с извлеченным содержимым прямо из меню главного окна программы;

- гибкие настройки: возможность указать место для распаковки архивов по умолчанию, активировать автоматическую распаковку при добавлении архива в программу, режим «поверх остальных окон», возможность сворачивания в трей и др.

Учитывая все это, а также бесплатность и небольшие размеры, ExtractNow является неплохим выбором. Единственный минус - этот архиватор не имеет русской локализации. Но, учитывая простоту программы, это не большая проблема. Заканчивать курсы английского языка для ее использования не обязательно, поскольку в интерфейсе все наглядно и интуитивно понятно.

**Литература (ссылки)**

[https://ru.wikipedia.org/wiki/Архиватор](https://m.vk.com/away?to=https%3A%2F%2Fru.wikipedia.org%2Fwiki%2F%D0%90%D1%80%D1%85%D0%B8%D0%B2%D0%B0%D1%82%D0%BE%D1%80)

[https://artursharipov.ru/landing/disc1/example/page1](https://m.vk.com/away.php?to=https%3A%2F%2Fartursharipov.ru%2Flanding%2Fdisc1%2Fexample%2Fpage1.html)

[https://www.softportal.com/software-2477-extractnow](https://m.vk.com/away?to=https%3A%2F%2Fwww.softportal.com%2Fsoftware-2477-extractnow)

<https://www.computerra.ru/263767/dlya-chego-nuzhny-arhivatory-fajlov>