Министерство образования и науки
Донецкой Народной Республики

ГОУВПО «Донецкий национальный технический университет»

Факультет «Компьютерные науки и технологии»

Кафедра «Программная инженерия»

Программная инженерия: методы и технологии разработки информационно-вычислительных систем

(ПИИВС-2018)

Сборник материалов II Международной научно-практической конференции

(студенческая секция)

г. Донецк
14-15 ноября 2018 года

Донецк – 2018

УДК 004.4
ББК 32.81
П78

Программная инженерия: методы и технологии разработки информационно-вычислительных систем (ПИИВС-2018): сборник научных трудов II научно-практической конференции (студенческая секция), Том 2, 14-15 ноября 2018 г. – Донецк, ГОУВПО «Донецкий национальный технический университет», 2018. – 264с.

Студенческая секция, проводимая в рамках научной конференции ПИИВС-2018, является одним из этапов совместной исследовательской деятельности преподавателей и студентов. Основная задача студенческой секции состояла в укрепление научного и педагогического сотрудничества среди студентов и научных руководителей, развитие у студентов навыков самостоятельной работы, способностей к анализу и обобщению изучаемого материала, умению формировать собственные выводы и заключения, излагать их письменно и в форме публичных выступлений.

Основные направления работы студенческой секции конференции:

- программная инженерия;
- системы автоматизированного проектирования;
- информационные технологии в медиаиндустрии и дизайне
- прикладная математика;
- информационные системы и технологии в технике и бизнесе;
- системный анализ и интеллектуальные информационные системы.

На 4 подсекциях были представлены 53 доклада, в которых рассмотрен широкий спектр научных направлений работ студентов.

Конференция организована Донецким национальным техническим университетом Министерства образования и науки ДНР. В организации конференции приняли участие: Донецкий национальный университет, Министерство связи ДНР, Ульяновский государственный технический университет (г.Ульяновск), Образовательно-научный центр «Кибернетика» ФГБОУ ВО «РЭУ им. Г.В. Плеханова» (г.Москва), Полоцкий государственный университет (Республика Беларусь, г.Полоцк).

Во втором томе сборника научных трудов представлены доклады студентов, которые учатся в бакалавриате и магистратуре высших учебных заведений из Ижевска, Кемерово, Таганрога и Донецка.

УДК 004.4
ББК 32.81

ОГЛАВЛЕНИЕ

СЕКЦИЯ «ПРОГРАММНАЯ ИНЖЕНЕРИЯ» **8**
Алексеев А.А. (Ижевск, Удмуртский государственный университет)
ПРИМЕНЕНИЕ МЕТОДА ХАФФМАНА ДЛЯ СЖАТИЯ ИНФОРМАЦИИ 8

Артеменко О.Г., Федяев О.И. (Донецк, ДонНТУ)
ПОВЫШЕНИЕ ИНФОРМАЦИОННО-ОБРАЗОВАТЕЛЬНОГО УРОВНЯ САЙТА КАФЕДРЫ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ 13

Варяник А.С., Грищенко В.И. (Донецк, ДонНТУ)
ЛЕКСИЧЕСКИЙ И СИНТАКСИЧЕСКИЙ РАЗБОР КАК ПЕРВЫЙ ЭТАП СТАТИЧЕСКОГО АНАЛИЗА SQL-КОДА 19

Вивденко В.С., Копытова О.М. (Донецк, ДонНТУ)
ОБЗОР АКТУАЛЬНЫХ МЕТОДОВ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ КОМНАТ В ВИДЕОИГРАХ 23

Воробьев Л.О., Григорьев А.В. (Донецк, ДонНТУ)
АНАЛИЗ СПЕЦИФИКИ РЕАЛИЗАЦИИ ОБЪЕКТНОГО ПОДХОДА В РАЗЛИЧНЫХ ТЕХНОЛОГИЯХ ПРОГРАММИРОВАНИЯ 27

Вязмин В.И., Чернышова А. В. (Донецк, ДонНТУ)
ГОЛОСОВЫЕ МЕССЕНДЖЕРЫ. ОБЗОР СУЩЕСТВУЮЩИХ ПРОТОКОЛОВ 33

Гончаров К.Д., Федяев О. И. (Донецк, ДонНТУ)
РЕЧЕВОЕ УПРАВЛЕНИЕ СИСТЕМОЙ ОЦЕНИВАНИЯ МЕНТАЛЬНОСТИ СТУДЕНТОВ 38

Григорьев А.В., Гурин А.Г. (Донецк, ДонНТУ)
АНАЛИЗ СОСТОЯНИЙ МЕТОДОВ, АЛГОРИТМОВ И ПРОГРАММНЫХ СРЕДСТВ В ПОДДЕРЖКЕ УПРАВЛЕНИЯ ВЕРСИЯМИ 43

Зарецкий А.О., Ложкин А.В. (Кемерово, Кузбасский государственный технический университет имени Т.Ф. Горбачёва (КузГТУ))
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И МОБИЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ КОНТРОЛЯ ЗА ПАТОГЕНЕЗОМ И ЛЕЧЕНИЕМ ГОЛОВНЫХ БОЛЕЙ 48

Жильцов В.А. (Донецк, ДонНТУ)
ПРОБЛЕМЫ МНОГОПОТОЧНОГО ПРОГРАММИРОВАНИЯ НА ПЛАТФОРМЕ .NET FRAMEWORK 52

Журавлёв А.В. (Донецк, ДонНТУ)
СОЗДАНИЕ ОБУЧАЮЩЕЙ СИСТЕМЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ PHP В ВИДЕ КВЕСТ-ИГРЫ 57

Кутелёв Р.С., Рычка О.В. (Донецк, ДонНТУ)
РАЗРАБОТКА БАЗЫ ДАННЫХ ДЛЯ ХРАНЕНИЯ ДОКУМЕНТОВ НА ПРЕДПРИЯТИИ .. 61

Макогон С.А., Зори С.А. (Донецк, ДонНТУ)
СРАВНЕНИЕ ПОПУЛЯРНЫХ БИБЛИОТЕК КОМПЬЮТЕРНОГО ЗРЕНИЯ ДЛЯ ИСПОЛЬЗОВАНИЯ В ПРИЛОЖЕНИИ ПО РАСПОЗНАВАНИЮ ТРАНЗИСТОРОВ 66

Махорин С.Н., Коломойцева И.А. (Донецк, ДонНТУ)
ПРИЛОЖЕНИЕ ДЛЯ ПОИСКА НА МЕСТНОМ РЫНКЕ УСЛУГ ОБЩЕСТВЕННОГО ПИТАНИЯ 71

Медведев А.С., Федяев О.И. (Донецк, ДонНТУ)
ЛОГИЧЕСКАЯ МОДЕЛЬ СВЁРТОЧНОЙ НЕЙРОННОЙ СЕТИ ДЛЯ РАСПОЗНАВАНИЯ ЛИЦА ЧЕЛОВЕКА 76

Московченко А.В., Жабская Т.Е., Федяев О.И. (Донецк, ДонНТУ)
ПРЕДСТАВЛЕНИЕ ВИРТУАЛЬНОЙ КАФЕДРЫ НА УРОВНЕ ВИЗУАЛЬНЫХ МОДЕЛЕЙ МНОГОАГЕНТНОЙ СРЕДЫ JACK 82

Ольшевский А.И., Нестеренко В.С. (Донецк, ДонНТУ)
СИСТЕМА УДАЛЕННОГО РЕЗЕРВНОГО КОПИРОВАНИЯ ДАННЫХ ДЛЯ КОРПОРАТИВНЫХ СЕТЕЙ 88

Поздняков А.А. (Донецк, ДонНУ)
ОСОБЕННОСТИ РЕАЛИЗАЦИИ ШИФРА ХИЛЛА В КОМПЬЮТЕРНЫХ ПРИЛОЖЕНИЯХ 92

Полетаев В.А., Коломойцева И.А. (Донецк, ДонНТУ)
КВАНТОВАНИЕ ЦВЕТОВОГО ПРОСТРАНСТВА В КОНТЕКСТЕ РЕШЕНИЯ ЗАДАЧИ ПОИСКА ИЗОБРАЖЕНИЙ ПО СОДЕРЖАНИЮ 96

Полищук С.Ю., Незамова Л.В. (Донецк, ДонНТУ)
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ АВТОМАТИЗАЦИИ ПРОИЗВОДСТВЕННОЙ ДЕЯТЕЛЬНОСТИ СЛУЖБЫ ОРГАНИЗАЦИИ ДОРОЖНОГО ДВИЖЕНИЯ 102

Ржевский К.В. (Донецк, ДонНТУ)
РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ И СОВРЕМЕННОЙ СИСТЕМЫ УЧЁТА И ЗАЩИТЫ ДАННЫХ В КОНТЕКСТЕ МАЛОГО ПРЕДПРИЯТИЯ ДЛЯ СКЛАДСКОЙ ДОКУМЕНТАЦИИ 108

Сидорчук В.И., Губенко Н.Е., Сипаков Д.С. (Донецк, ДонНТУ)
МЕТОД СОКРЫТИЯ СООБЩЕНИЙ НА ОСНОВЕ ЖАРГОНОВ 114

Толбатова А.С., Чернышова А.В. (Донецк, ДонНТУ)
ОБЗОР МЕТОДОВ АУДИО СТЕГАНОГРАФИИ 119

Хубеджев Д.П., Ситникова О.Д. (Донецк, ДонНТУ)
РАЗРАБОТКА ПО ОПТИМИЗАЦИИ ЛОГИСТИКИ ПОСТАВОК ПРОДУКЦИИ 123

Янкивский А.А., Павлова Е.М., Федяев О.И. (Донецк, ДонНТУ)
ПРОЕКТИРОВАНИЕ В СРЕДЕ MADKIT АГЕНТНО-ОРИЕНТИРОВАННОЙ СИСТЕМЫ ПРОГНОЗИРОВАНИЯ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ СТУДЕНТОВ 127

СЕКЦИЯ «СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ»..... 134
Бондаренко Е.С., Григорьев А.В. (Донецк, ДонНТУ)
АНАЛИЗ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ И ПЕРСПЕКТИВА РАЗВИТИЯ САПР КОРАБЛЕЙ..... 134

Горбань В.В., Григорьев А.В. (Донецк, ДонНТУ)
ОБЗОР СОВРЕМЕННЫХ ТЕХНОЛОГИЙ, ПРИНЦИПОВ И ПРОЦЕССОВ 3D-ПЕЧАТИ.. 140

Дубянская А.О., Григорьев А.В. (Донецк, ДонНТУ)
АНАЛИЗ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ СОВРЕМЕННЫХ СИСТЕМ УПРАВЛЕНИЯ «УМНЫМИ» ТЕПЛИЦАМИ И ПЕРСПЕКТИВЫ РАЗВИТИЯ 145

Капков Ю.Д., Григорьев А.В. (Донецк, ДонНТУ)
АНАЛИЗ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ И ПЕРСПЕКТИВА РАЗВИТИЯ САПР ПО СОСТАВЛЕНИЮ РАЦИОНА ПРАВИЛЬНОГО ПИТАНИЯ..... 149

Макорин С.А., Григорьев А.В. (Донецк, ДонНТУ)
АНАЛИЗ МЕТОДОВ И ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ РАЗРАБОТКИ ИГР 155

Мамутова В.А., Григорьев А.В. (Донецк, ДонНТУ)
ПРОБЛЕМАТИКА СОЗДАНИЯ ТРЁХМЕРНЫХ МОДЕЛЕЙ ГОРОДОВ И АНАЛИЗ РЫНКА ПРИЛОЖЕНИЙ ДЛЯ 3D-МОДЕЛИРОВАНИЯ..... 160

Назарко А.В., Григорьев А.В. (Донецк, ДонНТУ)
ОБЗОР И СРАВНЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ТРЕХМЕРНОГО МОДЕЛИРОВАНИЯ 166

Пыльцов Д.А., Григорьев А.В. (Донецк, ДонНТУ)
ОБРАЗОВАТЕЛЬНАЯ СОЦИАЛЬНАЯ СЕТЬ КАК СОВРЕМЕННЫЙ ПОДХОД К ОБУЧЕНИЮ..... 170

Семик А.О., Филипишин Д.А., Григорьев А.В. (Донецк, ДонНТУ)
АНАЛИЗ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ СОЗДАНИЯ МУЗЫКАЛЬНЫХ КОМПОЗИЦИЙ И ПЕРСПЕКТИВЫ ИХ РАЗВИТИЯ 175

Чернышов Д.Н., Григорьев А.В. (Донецк, ДонНТУ)
АНАЛИЗ ФУНКЦИЙ И ПЕРСПЕКТИВЫ РАЗВИТИЯ САПР ТРУБОПРОВОДОВ..... 182

СЕКЦИЯ «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В МЕДИАИНДУСТРИИ И ДИЗАЙНЕ»..... 187
Бобелюк М.Б., Губенко Н.Е. (Донецк, ДонНТУ)
ДОПОЛНЕННАЯ РЕАЛЬНОСТЬ КАК КОНЦЕПЦИЯ СОЗДАНИЯ ОБУЧАЮЩЕГО ПРИЛОЖЕНИЯ..... 187

Давыденко Д.П., Губенко Н.Е. (Донецк, ДонНТУ)
ОБУЧАЮЩАЯ СИСТЕМА С АДАПТАЦИЕЙ ПО ФОРМЕ ИЗЛОЖЕНИЯ МАТЕРИАЛА НА ОСНОВЕ МЕНТАЛЬНЫХ КАРТ..... 191

Дементьева Е.Е., Рейзенбук К.Э. (Кемерово, Кузбасский государственный технический университет имени Т.Ф. Горбачева)
АНАЛИЗ КОРПОРАТИВНОГО ПОРТАЛА С ЦЕЛЬЮ УЛУЧШЕНИЯ ПРОДУКТИВНОСТИ КОМПАНИИ 195

Кандаурова А.О., Губенко Н.Е. (Донецк, ДонНТУ)
АНАЛИЗ МОДЕЛЕЙ ИССЛЕДОВАНИЯ УДОВЛЕТВОРЕННОСТИ ПОТРЕБИТЕЛЕЙ МУЛЬТИМЕДИЙНОЙ ПРОДУКЦИИ 198

Кондратова Е.И., Киселёва О.В. (Донецк, ДонНТУ)
ПЛАНИРОВАНИЕ СОБСТВЕННОГО ИГРОВОГО ПРОЕКТА НА ОСНОВЕ АНАЛИЗА СУЩЕСТВУЮЩИХ ПРОЕКТОВ ПОДОБНОГО ЖАНРА И СТИЛЯ 203

Лащенова В.Э., Киселева О.В. (Донецк, ДонНТУ)
ОБЗОР ОНЛАЙН СИСТЕМ ДЛЯ РАЗВИТИЯ ЛОГИЧЕСКОГО МЫШЛЕНИЯ 208

Тилинина Н.Ю., Губенко Н.Е. (Донецк, ДонНТУ)
АНАЛИЗ ПРОБЛЕМ АРХИТЕКТУРЫ КОМПЬЮТЕРНЫХ ИГР 213

Юрченко А.С., Яшаров Д.А., Ефименко К.Н. (Донецк, ДонНТУ)
ОТДЕЛЬНЫЕ АСПЕКТЫ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ СИСТЕМ 218

СЕКЦИЯ «СИСТЕМНЫЙ АНАЛИЗ И ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ» 223
Shevlyakov A., Yefremchenko I., Gubenko N. (Donetsk, DonNTU)
APPLICATION OF DIGITAL WATERMARKS FOR DEVELOPMENT GRAPHIC PASSWORD SYSTEM 223

Горбенко Г.К., Караулов А.С. (Таганрог, Инженерно-технологическая академия ЮФУ)
КОРРЕКЦИЯ ОШИБОК ПРИ РАСПОЗНАВАНИИ РЕЧИ В МНОГОПОЛЬЗОВАТЕЛЬСКИХ СИСТЕМАХ 226

Ковалев Д.В., Копытова О.М. (Донецк, ДонНТУ)
РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДА ПОСТРОЕНИЯ ФУНКЦИИ ДЕГРАДАЦИИ ПОВЕДЕНИЯ КОНЕЧНОГО АВТОМАТА В РЕЗУЛЬТАТЕ ПЕРЕБРОСКИ ДУГ 230

Марченко В.В,. Ольшевский А.И. (Донецк, ДонНТУ)
ИССЛЕДОВАНИЕ АВТОМАТИЗИРОВАННЫХ ОБУЧАЮЩИХ СИСТЕМ В СФЕРЕ ОБРАЗОВАНИЯ 235

Минлигареев М.А., Ткаченко П.В. (Кемерово, Кузбасский государственный технический университет имени Т.Ф. Горбачева)
АНАЛИЗ ПРОБЛЕМЫ ТЕСТИРОВАНИЯ ИНЖЕНЕРНЫХ КАДРОВ 239

Нескородев Р.Н., Белик Т.С., Галиахметова К.Р. (Донецк, ДонНУ)
МЕТОДИКА ЭКСПЕРИМЕНТАЛЬНОГО ОПРЕДЕЛЕНИЯ ПОКАЗАТЕЛЕЙ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ 243

Оверченко Я.Ю., Копытова О.М. (Донецк, ДонНТУ)
ИССЛЕДОВАНИЕ РАЗРАБОТКИ ОНТОЛОГИЧЕСКОГО ЧАТ-БОТА НА БАЗЕ ГЛУБИННОГО ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ 249

Столбунская А.С., Кравец Т.Н. (Донецк, ДонНТУ)
СОЗДАНИЕ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ СТИЛИСТИЧЕСКОЙ ОЦЕНКИ ТЕКСТА 253

Строкин В.С., Ольшевский А.И. (Донецк, ДонНТУ)
ПРОЕКТИРОВАНИЕ ИНТЕЛЛЕКТУАЛЬНОГО ОБУЧАЮЩЕГО МОДУЛЯ «ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ» 257

Ткачёв Н.М., Федяев О.И. (Донецк, ДонНТУ)
ПАРАМЕТРИЧЕСКОЕ ОПИСАНИЕ МОДЕЛЕЙ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ В БИБЛИОТЕКЕ KERAS 261

СЕКЦИЯ «ПРОГРАММНАЯ ИНЖЕНЕРИЯ»

УДК 004.457

Применение метода Хаффмана для сжатия информации

Алексеев А.А.
Научный руководитель: Н.В. Латыпова
Удмуртский государственный университет
alekanton95@mail.ru

Алексеев А.А. Применение метода Хаффмана для сжатия информации. *В статье рассматривается алгоритм метода Хаффмана для сжатия информации, используется статическое и динамическое кодирование. С помощью написанной программы проведен сравнительный анализ файлов, содержащих различные данные и имеющих разные расширения. Сравнение проводилось как коэффициентом сжатия, так и по времени кодирования и раскодирования.*

Ключевые слова: *сжатие данных, алгоритм Хаффмана, дерево Хаффмана.*

Введение

Развитие телекоммуникационных систем, рост пропускной способности и общего количества линий связи, развитие глобальных компьютерных сетей и расширение спектра предоставляемых ими услуг с одновременным ростом числа пользователей компьютеров и сетей приводит к проблемам хранения и передачи различной информации (текстовой, звуковой, графической и т.п.). Применение сжатия происходит постоянно в любой области информационных технологий и не только. Например, радио и телевизор тоже используют сжатие, но только не данных, а сигнала.

Цель работы – реализация метода Хаффмана для сжатия данных и исследования его возможностей на файлах разных типов.

Нам потребуются следующие понятия определения:

1. Гнездо — это любой символ, т.е. гнездо является уникальным (неповторяющимся) для символов с различными кодами.
2. Частота появления символа — это сколько раз символ с данным кодом встречается в входном потоке.
3. Содержимое гнезда — это частота появления символа.
4. Пустое гнездо — когда содержимое гнезда равно нулю.
5. Статус гнезда — это состояние гнезда на текущий момент, т.е. гнездо может быть доступно (может участвовать при построении дерева) или недоступно (уже участвовало при построении дерева).
6. Двоичное дерево — это набор гнезд, соединенных некоторым способом между собой ветвями (связями). Каждое гнездо может иметь от одного до трёх связей, причем только одна ветвь может связывать с данное гнездо с гнездом более высшего уровня, но каждое гнездо может быть связано двумя ветвями с гнездами более низшего уровня (или не быть связанным вообще).

Классический алгоритм Хаффмана

Идея алгоритма состоит в следующем: зная вероятности 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 (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. 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.

Перед тем как начать сжатие потока данных, компрессор (кодер) должен построить коды. Это делается с помощью вероятностей (или частот появления) символов. Вероятности и частоты следует записать в сжатый файл для того, чтобы декомпрессор (декодер) Хаффмана мог сделать декомпрессию данных. Это легко сделать, так как частоты являются целыми числами, а вероятности также представимы числами. Обычно это приводит к добавлению нескольких сотен байтов в сжатый файл. Можно, конечно, записать в сжатый файл сами коды, однако это вносит дополнительные трудности, поскольку коды имеют разные длины. Еще можно записывать в файл само дерево Хаффмана, но это потребует большего объема, чем простая запись частот.

В любом случае, декодер должен прочесть начало файла и построить дерево Хаффмана для алфавита. Только после этого он может читать и декодировать весь файл. Алгоритм декодирования очень прост. Следует начать с корня и прочитать первый бит сжатого файла. Если это нуль, следует двигаться по нижней ветке дерева; если это единица, то двигаться надо по верхней ветке дерева. Далее читается второй бит и происходит движение по следующей ветке по направлению к листьям. Когда декодер достигнет листа дерева, он узнает код первого несжатого символа (обычно это символ ASCII). Процедура повторяется для следующего бита, начиная опять из корня дерева [1].

Статическое кодирование

Необходимые определения

1. Гнездо — это любой символ, т.е. гнездо является уникальным (неповторяющимся) для символов с различными кодами.

2. Частота появления символа — это сколько раз символ с данным кодом встречается в входном потоке.

3. Содержимое гнезда — это частота появления символа.

4. Пустое гнездо — когда содержимое гнезда равно нулю.

5. Статус гнезда — это состояние гнезда на текущий момент, т.е. гнездо может быть доступно (может участвовать при построении дерева) или недоступно (уже участвовало при построении дерева).

6. Двоичное дерево — это набор гнезд, соединенных некоторым способом между собой ветвями (связями). Каждое гнездо может иметь от одного до трёх связей, причем только одна ветвь может связывать с данное гнездо с гнездом более высшего уровня, но каждое гнездо может быть связано двумя ветвями с гнездами более низшего уровня (или не быть связанным вообще).

Описание алгоритма.

1. Если число доступных, непустых гнезд менее чем 2, то перейти к пункту 5.

2. Среди доступных гнезд ищем два гнезда с минимальным (не равным нулю) содержимым гнезда.

3. Создается новое гнездо с содержимым равным сумме содержимых двух найденных гнезд. Новому гнезду присваивается статус доступно, а двум найденным гнездам присваивается статус недоступны и дополнительный статус: первому найденному гнезду - 0, второму - 1.

4. Перейти к пункту 1.

5. Это одно оставшееся доступное ненулевое гнездо — мы назовем корень дерева и дополнительного статуса не имеет.

Теперь каждый символ из входного потока будет заменяться последовательностью бит, получаемую шагая от корня дерева по связям к гнезду, соответствующему кодируемому символу, и при этом запоминается дополнительный статус каждого из гнезд в этой цепочке. Сформированная цепочка бит и будет кодом текущего символа. Она имеет переменную длину (от 1 до 256 бит теоретически, практически же почти всегда до 12).

После того как мы закодировали текст, мы должны позаботиться, чтобы он мог быть раскодирован. Для этого распаковщику надо знать дерево, поэтому возникает необходимость сохранения дерева в файле, который содержит упакованный текст, и естественно при наименьшем размере.

Варианты записи дерева в файл:

I. Если взять и записать дерево не сжимая, то понадобится 256 последовательностей бит:

1) 4 бита — длина цепочки битов, являющейся кодом символа

2) Переменная длина (0..15) — сама последовательность бит

В самом худшем случае мы получим дерево, записанное в 256+128=384 байта, что является довольно большой потерей процента упаковки.

II. Теперь попробуем усовершенствовать предыдущий способ. Так как мы знаем, что длина кода практически никогда не превышает 12 бит, то длины последовательностей равные 13,14,15 мы можем использовать в качестве управляющих. Определим:

1) управляющий код 14 при первом появлении — указывает что следующие 8 бит являются кодом первого используемого символа, имеющего наименьший код ASCII (это означает, что если к примеру, использован текстовый файл для упаковки, то наименьший используемый код в тексте имеет возврат каретки (код=13), символы же с кодами меньше 13 не используются в данном тексте и поэтому нет смысла запоминать их в при

упаковке дерева. Иными словами, это означает, что символы с кодами меньше 13 при постройке дерева не участвовали).

2) управляющий код 15 — префикс повторения, означает что следующие 4 бита являются числом 4-х битовых нулевых последовательностей. Позволяет группировать от 3 до 18 таких последовательностей.

3) управляющий код 13 — префикс повторения, означает, что следующие 4 бита, являются числом 4-х битовых нулевых последовательностей. Позволяет группировать от 19 до 34 таких последовательностей.

4) управляющий код 14 при втором появлении означает конец дерева, то есть что символы с кодами больше, чем последнее записанное значение не участвовали при построении дерева.

III. Третий вариант кодирования дерева:

1 байт — количество вершин в дереве;

2 байт — символы являющиеся вершинами дерева (кол-во указано в первом байте)

3 байт — последовательности битов, означающие:

1) Четыре бита - количество бит на символ (0000=1 .. 1111=16)

2) Четыре бита - сколько символов с указанной выше битовой длиной (0000=1 .. 1110=15), если 1111 - то считать еще 5 бит. Пять бит - количество символов (00000=16 .. 11110=46), если 11111 - то считать еще 5 бит и т.д. (количество таких сочетаний указано в первом байте).

4 бит последовательности бит, означающие битовые последовательности, означающие Хаффмановский код символа в том порядке, как они приведены выше [2].

Динамическое кодирование

Динамическое кодирование имеет преимущество по сравнению со статическим кодированием Хаффмана: оно хорошо приспосабливается к быстро меняющимся данным и нет необходимости хранить дерево вместе с запакованным тестом.

Первоначально дерево выглядит так: все символы ASCII имеют частоту появления, равную единице, и строится дерево. Читается символ из входного потока. Частота соответствующего символа в дереве увеличивается на 1 и дерево перестраивается. По мере того, как некоторое количество символов считается из входного потока, и по мере того, как часто перестраивается дерево, символы, встречающиеся чаще, будут иметь меньшую длину кода, чем остальные.

Рассмотрим описание метода на основе примера:

"маша_мыла_раму,_мама_мыла_тоже"

Подсчитаем частоты появления символов:

'а' - 7, 'м' - 6, '_' - 5, 'ы' - 2, 'л' - 2, 'р' - 1, 'у' - 1, ',' - 1, 'т' - 1, 'о' - 1, 'ж' - 1, 'е' - 1.

Дерево запакуется: (по байтам) 12 (количество символов - 1), 'а','м','_','ы','л','ш','р', 'у',',','т','о','ж','е', (сами символы) $11 (шестнадцатеричное. Длина кода = 7-1=6), $20, $31, $47, (далее идут Хаффмановские коды символов)

|00 01 100 1|010 1011 1|1000 1100|1 11010 11|011 11100| 11101 111|10 11111

Знак "|" делит битовые последовательности на байты.

Дерево пакуется в 199 бит (24 байта и 7 бит)

Далее идет запакованный текст:

01 00 11000 00 100 01 1010 1011 00 100 11001 00 01 11010 11011 100

(м а ш а _ м ы л а _ р а м у , _)

01 00 01 00 100 01 1010 1011 00 100 11100 11101 11110 11111

(м а м а _ м ы л а _ т о ж е)

Всего: 12 байт и 1 бит = 97 бит.

Общая длина с деревом: 38 байт.[3]

Анализ возможностей программы

По данному алгоритму была написана программа на языке C# в среде разработки Visual Studio 2013, и с помощью неё проведен сравнительный анализ по сжатию (и восстановлению) файлов с различными данными (и разными расширениями) с помощью написанной программы.

Таблица 1 – Коэффициент сжатия

Тип файла	Формат данных	Исходные данные (в КБ)	Сжатые данные (в КБ)	Коэффициент сжатия
docx	текст	31	32	1,1
doc	текст	6161	3111	0,5
max	3D-изображение	200	95	0,475
mkv	видео	666540	664589	0.99
pptx	презентация	317	308	0.97

ppt	презентация	1109	1000	0.9
jpg	картинка	619	620	1
bmp	картинка	6751	6499	0.96
gif	картинка	984	985	1
png	картинка	3188	3188	1
txt	текст	420	226	0,54
png	Черно-белая картинка	1387	1388	1,01
jpg	Черно-белая картинка	778	778	1
txt	Числа	1427	647	0,45
rtf	текст	11888	5488	0,46

Таблица 2 – Время сжатия и раскодирования

Тип файла	**Формат данных**	**Время сжатия (в сек)**	**Время раскодирования (в сек)**
docx	текст	1	0,1
doc	текст	1	0,2
max	3D-изображение	1.5	0,5
mkv	видео	941	120
pptx	презентация	1.5	0,1
ppt	презентация	2	0,1
jpg	картинка	1	0,1
bmp	картинка	10	2
gif	картинка	2	1
png	картинка	6	0,1
txt	текст	1	0,1
png	Черно-белая картинка	2	0,1
jpg	Черно-белая картинка	1	0,1
txt	Числа	1	0,1
rtf	текст	10	0,1

Как видно из таблицы 1, в расширениях .docx, .mkv, .pptx, .ppt, .jpg, .bmp, .gif, .png коэффициент сжатия близок или больше 1. Связано это, скорее всего, с тем, что в файлы с такими расширениями сжатие уже заложено изначально.

Из таблицы 1 видно, что хороший коэффициент сжатия (около 0,5) получается для файлов с расширениями .doc,. max, .txt, .rtf. Для таких расширений можно использовать сжатие по методу Хаффмана.

В таблице 2 представлено сравнение между временем сжатия и раскодирования. Восстановление происходит без потери данных, лишь необходимо знать тип файла, т.к. программа не запоминает тип сжатого файла. По времени восстановление происходит гораздо быстрее, чем сжатие.

Выводы

В работе представлена реализация метода Хаффмана на языке C# и исследованы возможности полученного программного продукта. Сравнительный анализ проводится на файлах, содержащих данные различной природы и имеющих разные расширения. Лучший коэффициент сжатия (≈0,5) получается для файлов с расширениями .doc,. max, .txt, .rtf. Восстановление происходит без потерь и быстрее, чем сам процесс сжатия.

Литература

1. Ватолин Д.С. Алгоритмы сжатия изображений / Метод.пособие. М., 1999. 76 с.
2. Кормен Т. Алгоритмы: построение и анализ // Т. Кормен, Ч. Лейзерсон, Р.Ривест, К.Штайн – Москва: Вильямс, 2012. – 1296 с.
3. Дьяков А. Метод сжатия Хаффмана 2010 – Режим доступа: http://www.codenet.ru/progr/alg/huffman/
4. Верещагин Н.К. Колмогоровская сложность и алгоритмическая случайность // Н.К. Верещагин, В.А. Успенский, А. Шень – Москва: МЦНМО, 2013. – 575 с.
5. Сэломон Д. Сжатие данных, изображений и звука Москва: Техносфера, 2004. 368 с.

Алексеев А.А. Применение метода Хаффмана для сжатия информации. *В статье рассматривается алгоритм метода Хаффмана для сжатия информации, используется*

статическое и динамическое кодирование. С помощью написанной программы проведен сравнительный анализ файлов, содержащих различные данные и имеющих разные расширения. Сравнение проводилось как коэффициентом сжатия, так и по времени кодирования и раскодирования.

Ключевые слова: *сжатие данных, алгоритм Хаффмана, дерево Хаффмана.*

Alekseev Anton. The application of the Huffman method to compress the information. *The article discusses the Huffman algorithm to compress the information used for static and dynamic encoding. With the help of the written program, a comparative analysis of files containing different data and having different extensions is carried out. The comparison was carried out both by compression ratio and by encoding and decoding time.*

Keywords: *data compression, Huffman algorithm, Huffman tree.*