Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовая работа по курсу «Дискретный анализ»: Методы сжатия данных

Студент: О. Р. Лисовский
Преподаватель: Н. А. Зацепин
Группа: М8О-408Б
Дата:
Оценка:
Подпись:

Москва, 2020

Условие

Необходимо реализовать два известных метода сжатия данных для сжатия одного файла.

Формат запуска должен быть аналогичен формату запуска программы gzip, должны быть поддержаны следующие ключи: -c, -d, -k, -l, -r, -t -1 -9. Должно поддерживаться указание символа дефиса в качестве стандартного ввода.

Метод решения

Как и требуется в условии запуск программы аналогичен запуску утилиты gzip: ./main <ключи> <файлы> <ключи> <файлы> ...

Обработка входных данных

Программа начинается с обработки строки стандартного ввода. Строка обрабатывается по словам. Если слово начинается с символа «-», то предполагается что это набор ключей и ключи передаются в специальную функцию, чтобы исключить противоречия работы ключей. Возможные сочетания и глвенство одних ключей над другими описано в таблице ниже.

	d	k	l	r	t	1	9
c	нет блока	c блокирует k	l блокирует c	нет блока	t блокирует c	нет блока	нет блока
d	-	нет блока	l блокирует d	нет блока	t блокирует d	d блокирует 1	d блокирует 9
k	-	-	l блокирует k	нет блока	t блокирует k	нет блока	нет блока
l	-	-	-	нет блока	l блокирует t	l блокирует 1	l блокирует 9
r	-	-	-	-	нет блока	нет блока	нет блока
t	-	-	-	-	-	t блокирует 1	t блокирует 9
1	-	-	-	-	-	-	последний полученный блокирует прошлые

В случае если полученное слово начинается с другого символа, то программа предполагает что это имя файла или директории и добавляет его в список для дальнейшей обработки.

Интерфейс

После установления активных ключей и заполнения списка объектов компрессии/-декомпрессии, программа начинает работу этим самым списком. В случае отсутствия ключа -r все директории не рассматриваются. При активации ключа всё содержимое директории рекурсивно обрабатывается программой. При работе с файлами проверяется наличие/отсутствие (в зависимости от ключа -d) файла с расширением .gz и в случае необходимости программа спрашивает у пользователя право на перезапись соответствующего файла.

При подготовке непосредственно компрессии проверяется наличие ключа -1 или -9 для определения необходимого алгоритма. В случае их отсутствия используются оба алгоритма и выбирается лучший результат.

При подготовке непосредственно декомпрессии читается первый байт файла для установления алгоритма декодирования.

Постобработка

При окончании работы компрессии/декомпрессии программа получает сигнал об их завершении. Если этот сигнал соответствует ошибке то работа с конкретным файлом аварийно прекращается и обрабатывается следующий файл. Дальнейшие действия обусловлены введёнными ключами.

Далее незакодированный файл будет упоминаться как файл, а закодированный файл как архив.

Арифметическая компрессия

Арифметическая декомпрессия

LZ77 компрессия

LZ77 декомпрессия

По окончании чтения архива, количество байт, которое было в изначальном файле, сверяется с тем, сколько было записано в его новую версию. При несовпадении выводится соответствующее сообщение, и декомпрессия завершается неудачно.

Описание файлов программы

Код программы разбит на 9 файлов:

1. ACC.h - Содержит базовую информацию о классе TACC, необходимом для работы компрессии и декомпрессии соответствующего алгоритма.

2. ACC.cpp - Содержит реализацию класса TACC.

3. BFile.h - Содержит базовую информацию о классах TOutBinary и класса TInBinary, необходимых для работы с файлами.
4. BFile.cpp - Содержит реализацию классов TOutBinary и TInBinary.
5. interface.h - Содержит в себе перечисление и описание всех функций необходимых для взаимодействия программы и алгоритмов сжатия данных.
6. interface.cpp - Содержит реализацию всех функций, описанных в файле interface.h.
7. Library.h - Содержит в себе ключи, необходимые для работы алгоритмов, и библиотеки для работы всей программы.
8. LZ77.h - Содержит базовую информацию о классе TLZ77, необходимом для работы компрессии и декомпрессии соответствующего алгоритма.
9. LZ77.cpp - Содержит реализацию класса TLZ77.
10. main.cpp - Файл запуска.
11. Makefile - Сборочный файл.

Основные типы данных

1. TOutBinary - класс, обеспечивающий запись необходимого количества байт в файл.
2. TInBinary - класс обеспечивающий считывание необходимого количества байт из файла.
3. TLZ77 - класс, описывающий работу алгоритма LZ77.
4. TACC - класс, описывающий работу арифметического алгоритма.

Описание методов и функций программы

Основные свойства и методы класса TACC

public:

1. bool Compress (const char*, const char*) - сжатие файла;
2. bool Decompress (const char*, const char*) - распаковка файла;
3. TACC() - конструктор, в котором задаются начальные значения для последующей работы со сжатием/распаковкой файла;

private:

1. bool chError - флаг ошибки при распаковке файла;
2. unsigned char indexToChar [NO_OF_SYMBOLS] - таблица перевода из индексов к символам;
3. int charToIndex [NO_OF_CHARS] - таблица перевода из символов в индексы;
4. int cumFreq [NO_OF_SYMBOLS + 1] - массив накопленных частот. Нужен для определения границ;
5. int freq [NO_OF_SYMBOLS + 1] - массив частот. В нём хранится число появлений тех или иных символов;
6. long low - нижняя граница отрезка;
7. long high - верхняя граница отрезка;
8. long value - число, которое лежит в отрезке;
9. long bitsToFollow - количество бит, которые надо пустить в след за следующим выставляемым битом;
10. int buffer - буффер для работы с файлом;
11. int bitsToGo - число битов, которые ещё можно загрузить в буффер;
12. int garbageBits - счётчик плохих битов при распаковке файла. Как только их становится слишком много - распаковка отменяется и выводится сообщение об этом;
13. FILE *out - файл, в который мы записываем;
14. FILE *in - файл, из которого мы считываем;
15. void UpdateModel (int) - обновление модели под новый символ;
16. void EncodeSymbol (int) - кодировка символа;
17. void InputFileInfo() - запись информации о сжимаемом файле;
18. void StartEncoding() - подготовка к сжатию;
19. void DoneEncoding() - завершение кодирования. Загрузка последних битов в буффер;
20. void StartDecoding() - подготовка к распаковке;
21. int DecodeSymbol() - распаковка символа;
22. int InputBit() - получение одного бита из файла;
23. void OutputBit(int) - отправление одного бита в файл;
24. void OutputBitPlusFollow(int) - вывод указанного бита и отложенных ранее;

Основные свойства и методы класса TOutBinary

public:

1. TOutBinary() - задаёт начальные значения. Файл не будет открыт.
2. bool Open(std::string*) - открывает файл;
3. bool Close() - закрывает файл;
4. bool Write(const char*, size_t) - запись в файл;
5. bool WriteBin(size_t bit) - запись бита в файл;
6. unsigned long long SizeFile() - подсчёт размера файла;
7. friend bool operator « (TOutBinary& file, size_t const &bit) - запись бита в файл;

private:

1. std::ofstream out - файл вывода;
2. std::string name - имя файла;
3. unsigned char head - маска для заноса бита в block;
4. unsigned char block - временный буффер для хранения и записи битов в файл;

Основные свойства и методы класса TInBinary

public:

1. TInBinary() - задаёт начальные значения. Файл не будет открыт.
2. bool Open(std::string*) - открывает файл;
3. bool Close() - закрывает файл;
4. bool Read(char*, size_t) - считывает из файла некоторое количество байт;
5. bool ReadBin(char* bit) - считывает из файла один бит;
6. unsigned long long SizeFile() - подсчёт размера файла;
7. friend bool operator » (TInBinary& iFile, char &bit) - получение бита из файла;

private:

1. std::ifstream in - файл вывода;

2. std::string name - имя файла;
3. unsigned char head - маска для заноса бита в block;
4. unsigned char block - временный буффер для хранения битов из файла, через него получают биты;

Основные свойства и методы класса TLZ77

public:

1.
2.
3.
4.
5.
6.
7.
8.

private:

1.
2.
3.
4.
5.
6.
7.
8.

Прочие функции

1. void FileIterator(std::map<std::string, int>) - Осуществляет проход по всем папкам и файлам для их компрессии/декомпрессии.
2. bool Parser(std::map<std::string, int>*, std::string) - фильтрует полученные при вводе аргументы. При получении некорректного аргумента возвращает false.
3. bool AskDir(std::string, bool) - Проверка на существование директории. При существовании возвращает true, в любом ином случае false.
4. void DirectoryWork(std::string) - В случае наличия ключа -r осуществляет работу с внутренними файлами и директориями указанной директории.
5. void DeComPress(std::string) - Помогает определить действия по отношению к указанному файлу: совершить компрессию, декомпрессию или посмотреть информацию об архиве.
6. bool Rewrite(std::string) - В случае возможного повторения имён файлов при компрессии/декомпрессии принимает решение о перезаписи.
7. void ErrorNotes(std::string) - Показывает сообщения об ошибках, возникших при работе с указанной директорией.
8. bool KeyL(TInBinary*, std::string) - Осуществляет работу ключа -l - вывод информации об архиве.
9. void PreCompress(TInBinary*, std::string) - Осуществляет подготовку указанного файла к сжатию в соответствии с указанными ключами.
10. unsigned long long int Compress(std::string, TInBinary*, bool) - Непосредственно активирует указанный алгоритм сжатия. Возвращает размер полученного архива или 0 в случае ошибки.
11. void PreDecompress(TInBinary*, std::string) - Осуществляет подготовку указанного файла к разжатию в соответствии с указанными ключами.

Исходный код

Тест производительности

Файл	Размер исходного файла (B)	Алгоритм	Время сжатия (c)	Время декомпрессии (c)	Размер сжатого файла	Коэффициент сжатия
world95.txt	3005020	LZ77				
world95.txt	3005020	Арифметика	1.3	1.5	1917592	1.6
world95.txt	3005020	оба				
world95.txt	3005020	gzip	0.4	2.5	878248	3.4
enwik8	100000000	LZ77				
enwik8	100000000	Арифметика	43.3	49.3	62762905	1.6
enwik8	100000000	оба				
enwik8	100000000	gzip	11.2	3.8	36518329	2.7
enwik9	1000000000	LZ77				
enwik9	1000000000	Арифметика	439.9	543.5	635524001	1.6
enwik9	1000000000	оба				
enwik9	1000000000	gzip	99.1	31.8	323742886	3,1

- Центральный процессор - Mobile DualCore Intel Celeron 1017U, 1600 MHz (16 x 100)
- Графический адаптер - Intel(R) HD Graphics (834742 KB)
- Оперативная память - DDR3-1333 DDR3 SDRAM 2 GB

Выводы

В процессе выполнения данной работы я освоил 2 вида кодирования: арифметическое и LZ77. Было обнаружены как сходства, так и различия. К примеру

Благодаря освоению двух алгоритмов сразу у меня появились представления о рабоче прочих алгоритмов кодирования и стали очевидны различные требования к их работе и результату. Были существенно улучшены навыки работы с файлами: проверка наличия, запись, чтение, перепись.

Список литературы

1. Алгоритм LZ77 [Электронный ресурс]: mf.grsu.by URL: http://mf.grsu.by/UchProc/livak/po/comprsite/theory_lz77.html (дата обращения 10.08.2020)
2. Алгоритмы LZW, LZ77 и LZ78 [Электронный ресурс]: habr.com URL: https://habr.com/ru/post/132683/ (дата обращения 23.08.2020)

3. Арифметическое кодирование [Электронный ресурс]: mf.grsu.by URL: http://mf.grsu.by/UchProc/livak/po/comprsite/theory_arithmetic.html (дата обращения 30.08.2020)

4. Идея арифметического кодирования [Электронный ресурс]: algolist.ru URL: http://algolist.ru/compress/standard/arithm.php (дата обращения 02.09.2020)

5. Arithmetic coding - integer implementation [Электронный ресурс]: stringology.org URL: http://www.stringology.org/DataCompression/ak-int/index_en.html (дата обращения 26.09.2020)