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: А. М. Титеев
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> ...

Препроцессинг

На первом этапе работы программа определяет наличие в поступившей строке ключей, директорий и файлов. При обработке ключей учитывается их взаимоперекрываение, как в утилите gzip: l и r имеют наибольший приоритет, далее идёт ключ t, после чего остальные. В случае, если новый ключ перекрывает по логике утилиты некоторые из уже имеющихся, то эти ключи деактивируются.

Если полученное слово из стандартного ввода не является ключом, то программа проверяет наличие директории с таким именем. Если такой директории нет, то считается, что это имя файла, и оно заносится в список файлов. Если директория с таким именем существует и подключён ключ r, то все файлы внутри этой директории добавляются в список.

После с файлами ведётся работа согласно введённым ключам.

Арифметическое кодирование

LZ77

Описание файлов программы

Код программы разбит на 9 файлов:

1. Arithmetic.h - Описывает класс Arithmetic, в котором заключён соответствующий алгоритм кодирования.
2. Arithmetic.cpp - Реализует класс Arithmetic.
3. Globals.h - Файл с общими библиотеками.
4. LZ77.h - Описывает класс LZ77, в котором заключён соответствующий алгоритм кодирования.
5. LZ77.cpp - Реализует класс LZ77.

6. main.cpp - Основной файл, отвечающий за чтение входных данных и принятие действий относительно каждого файла.
7. Makefile - Файл для сборки программы.
8. preprocessing.h - Содержит прототипы функций, необходимых для обработки данных перед компрессией/декомпрессией.
9. preprocessing.cpp - Реализует все функции из соответствующей библиотеки.

Основные типы данных

1. Arithmetic - Реализует соответствующий алгоритм.
2. LZ77 - Реализует соответствующий алгоритм.

Описание методов и функций программы

Основные свойства и методы класса Arithmetic

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 буфер.
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 StartInputingBits() - подготовка к побитовому вводу.
17. void StartOutputingBits() - подготовка к побитовому выводу.
18. void EncodeSymbol (int) - кодировка символа.
19. void StartEncoding() - подготовка к сжатию.
20. void DoneEncoding() - завершение кодирования. Загрузка последних битов в буфер.
21. void StartDecoding() - подготовка к распаковке.
22. int DecodeSymbol() - распаковка символа.
23. int InputBit() - получение одного бита из файла.
24. void OutputBit(int) - отправление одного бита в файл.
25. void DoneOutputingBits() - отправление последних битов в файл.
26. void OutputBitPlusFollow(int) - вывод указанного бита и отложенных ранее.

Основные свойства и методы класса LZ77

public:

1.
2.
3.

4.

5.

6.

7.

8.

private:

1.

2.

3.

4.

5.

6.

7.

8.

Прочие функции

1. void Compress(std::string) - Создает необходимые для компрессии данные (временные файлы).
2. void Decompress(std::string) - Создает необходимые для декомпрессии данные (классы и временные файлы).
3. bool ActivateKeys(std::string) - Активирует или деактивирует указанные в аргументах ключи. Из-за некорректного ключа функция возвращает false и завершает работу программы.
4. void ArchiveInfo(std::string) - Выводит информацию об архиве.
5. unsigned long long int CompressA(std::string) - Создаёт класс для арифметического кодирования и работает с ним. При некорректной работе алгоритма возвращает 0, иначе количество байт в получившемся файле.
6. unsigned long long int CompressL(std::string) - Создаёт класс для кодирования LZ77 и работает с ним. При некорректной работе алгоритма возвращает 0, иначе количество байт в получившемся файле.

7. bool ArchiveCheck(std::string) - Проверка на наличие у файла суффикса «.gz».
8. bool DirectoryCheck(std::string, bool) - Проверка на работоспособность директории.
9. void GetFiles(std::string, std::map<std::string, int>*) - Записывает все имена файлов директории в отдельное красно-чёрное дерево.
10. void SaveBest(std::string, unsigned long long int, unsigned long long int) - При отсутствии ключей 1 и 9 выбирает наименьший из полученных компрессией файлов и удаляет наибольший.
11. void ShowErrors(std::string) - Вывод сообщений о возникших ошибках во время подготовки данных для алгоритмов.
12. void Mv(std::string, std::string) - Выполняет команду mv для переименования временного файла.
13. void Rm(std::string) - Выполняет команду rm для удаления файла получившегося в процессе компрессии/декомпрессии и содержащего битые данные из-за ошибки алгоритма или удаляет поступивший алгоритму файл в случае отсутствия ключа k.

Исходный код

Тест производительности

В качестве теста производительности использовался файл размером, наполненный случайными символами английского алфавита.

Файл	Размер исходного файла	Алгоритм	Время сжатия (с)	Время «разжатия» (с)	Размер сжатого файла	Коэффициент сжатия
world95.txt						
enwik8						
enwik9						

Выводы

Благодаря выполнению данного проекта я научился основным принципам работы с файлами и директориями во время компрессии и декомпрессии. Так же я освоил основные правила и принципы работы компрессии и декомпрессии данных. Два построенных алгоритма дали мне необходимый базис навыков для работы с кастомными буферами. Так же я значительно улучшил свои навыки написания комплексных программ построения утилит, к примеру я научился реализовывать поддержку ключей и возможность работы нескольких алгоритмов.

Список литературы

1. Арифметическое кодирование - Arithmetic coding [Электронный ресурс]: ru.qwe.wiki URL: https://ru.qwe.wiki/wiki/Arithmetic_coding (дата обращения 28.08.2020)

2. Arithmetic Coding [Электронный ресурс]: users.cs.cf.ac.uk URL: https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node213.html (дата обращения 16.09.2020)

3. LZ77 на C, реализация алгоритма LZ77 на C [Электронный ресурс]: algor.skyparadise.org URL: https://algor.skyparadise.org/read/14 (дата обращения 16.09.2020)