Курсовой проект по курсу дискретного анализа: LZ77

Выполнил студент группы М8О-308Б-21 *Армишев Кирилл.*

Вариант: 1.5 Архиватор

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

1. Кодирование по Хаффману
2. LZ77

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

Метод решения

Для начала рассмотрим методы сжатия LZ77 и коды Хаффмана по отдельности.

LZ77:
В кодируемых строках часто содержатся совпадающие длинные подстроки. Идея, лежащая в основе LZ77, заключается в замене повторений на ссылки на позиции в тексте, где такие подстроки уже встречались.
Информацию о повторении можно закодировать парой чисел — смещением назад от текущей позиции (offset) и длиной совпадающей подстроки (length). В таком случае, например, строка pabcdeqabcde может быть представлена как pabcdeq⟨6,5⟩. Выражение ⟨6,5⟩ означает «вернись на 6 символов назад и выведи 5 символов».
Алгоритм LZ77 кодирует ссылки блоками из трёх элементов — ⟨offset,length,next⟩. В дополнение к двум уже описанным элементам, новый параметр next означает первый символ после найденного совпадающего фрагмента. Если LZ77 не удалось найти совпадение, то считается, что offset=length=0.

Коды Хаффмана:
Коды Хаффмана - это алгоритм сжатия данных, который используется для построения оптимальных префиксных кодов для минимизации общей длины закодированных данных. Этот алгоритм основан на частоте встречаемости символов в сообщении: символы с более высокой частотой получают более короткие коды, а символы с более низкой частотой получают более длинные коды.

Шаги алгоритма:

1. Нахождение частот символов: Пройти по тексту и подсчитать частоты каждого символа

2. Построение дерева Хаффмана: Добавляем символы с частотами, например, в priority_queue. Достаем два минимальных символа, объединяем их и добавляем обратно в priority_queue, суммируя частоты, так пока не останется одного символа X.

3. Присвоение кодов: Строим код для каждого символа на основе пути к нему в дереве. Путь к символу определяется по порядку прохождения от корня до листа. Для каждого символа в дереве, идя от корня к листьям: Если переходим к левому потомку добавляем бит "0 к правому - "1".

4. Кодирование сообщения: Проходим по каждому символу в сообщении и заменяем его соответствующим кодом.

5. Декодирование сообщения: Для декодирования необходима модель. Ее можно сериализовать и добавить в начало закодированного дерева. Десериализуем модель, строим дерево. Начинаем с корня и двигаемся вниз по дереву, используя биты закодированного сообщения. Когда достигнут лист, выводим соответствующий символ и возвращаемся к корню.

Необходимо сделать программу, подобную gzip. Рассмотрим значения флагов:

- -c - Выводит результат сжатия на стандартный вывод. Без этого флага, результат сжатия записывается в файл
- -d - Декодирует файл
- -k - Сохраняет оригинальный файл после сжатия или декомпрессии. Без этого флагафлага оригинальный файл удаляется
- -l - Выводит информацию о сжатом файле, такую как размер до и после сжатия, коэффициент сжатия
- -r - Рекурсивно обрабатывает все файлы в указанном каталоге.
- -t - Проверка целостности сжатия файла
- -1, -9 - Установить уровень сжатия. -1 обозначает наименьший уровень сжатия (быстрый), а -9 - максимальный уровень сжатия (медленный, но с более высоким коэффициентом сжатия).

Для более оптимального сжатия, я изначально применял алгоритм LZ77, получал триплеты, затем их сжимал с помощью кодов Хаффмана.

Описание программы

Для считывания опций использовался getopt. Относительно введенных опций заполняется структура options.

Кодирование:

1. Сжатие с помощью LZ77:
   В зависимости от введенных флагов(-0 или -9) устанавливается размер буфера(10000 или 500 символов). В зависимости от длины буффера зависит скорость и степень сжатия. Текст сжимается в триплеты, которые записываются во временный файл *buffer.txt*. Этот файл передается на сжатие в коды Хаффмана.

2. Коды Хаффмана:
   Функция *get_frequency_from_file()* возвращает *unordered_map<char, int>*, в котором хранятся символы и их частоты. По нему строится дерево. В итоговый файл записывается сериализованное дерево и закодированный текст. Для уменьшения объема двоичный закодированный код заменяется на char, т.е. 8 бит заменяются одним char-ом.

Декодирование

1. Коды Хаффмана:
   Десериализуем дерево, инвертируем char-ы обратно в 8-битные коды. Декодируем их в соответствии с алгоритмом во временный файл buffer.txt.

2. LZ77: Декодируем триплеты в файл.

Описаниее флагов

- -c - Выводит результат сжатия на стандартный вывод. Без этого флага, результат сжатия записывается в файл
- -d - Декодирует файл
- -k - Сохраняет оригинальный файл после сжатия или декомпрессии. Без этого флагафлага оригинальный файл удаляется
- -l - Выводит информацию о сжатом файле, такую как размер до и после сжатия, коэффициент сжатия
- -r - Рекурсивно обрабатывает все файлы в указанном каталоге.
- -t - Проверка целостности сжатия файла
- -1 - Устанавливает максимальный размер буффера - 250

- -9 - Устанавливает максимальный размер буффера - 10000(установлено по умолчанию)
- -0 - При кодировании исключается кодирование LZ77, применяются только коды Хаффмана

Пример работы программы

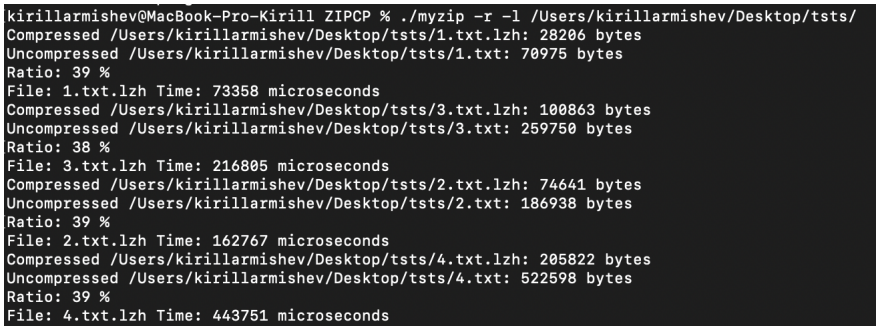


Рис. 1: Пример работы программы: Комбинация флагов -r -l

Дневник отладки

1. Кодирование Хаффманом не сжимало файл, а значительно увеличивало его размер.

   Решение: заменил двоичный код на char-овые символы, то есть проходя по тексту, архиватор заменяет 8 бит на один char.

2. Длительное время работы LZ77

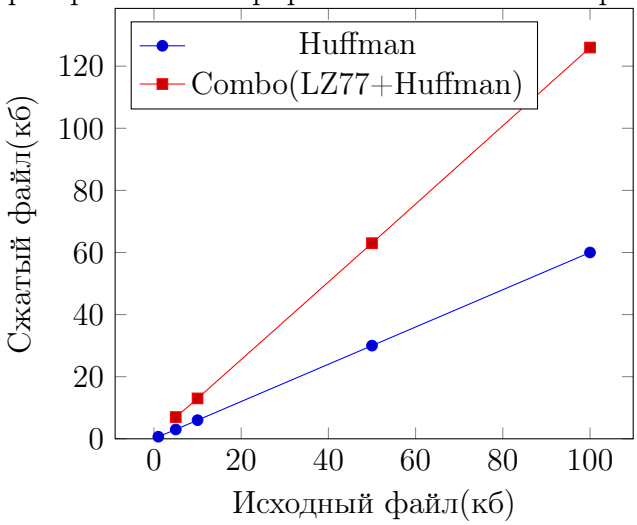
   Ошибка: Весь текст из исходного текста считывал в std::string, затем уже работал с ним.

   Решение: Считываю только кол-во символов равное максимальной размерности буффера. Время работы ускорилось в два раза.

Тест производительности

Сравнение степени сжатия файлов только алгоритмом LZ77 или комбинацией LZ77 и кодов Хаффмана:

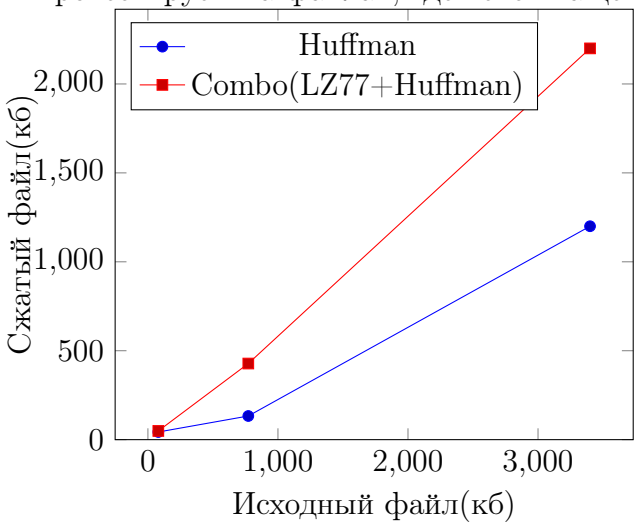
1. Проверим на сгенерированных текстовых файлах со случайными символами.



Размер исходного файла(кб)	Huffman(кб)	Combo(LZ77+Huffman)(кб)
1	0,672	2
5	3	7
10	6	13
50	30	63
100	60	126

Результаты логичны, потому что последовательность символов в данных файлах практически уникальна. Следовательно, использование комбинации LZ77+Huffman приведет к плохому сжатию, из-за того что мы увеличим объем исходного файла в три раза после применения LZ77.

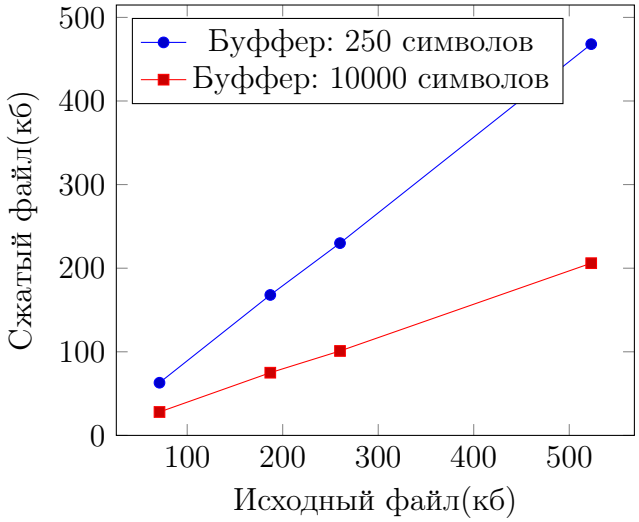
2. Протестируем на файлах, где текст чаще всего повторяется:



Тип файла	Размер исходного файла(кб)	Huffman(кб)	Combo(LZ77+Huffman)(кб)
C++	80	43	49
Python	772	133	428
LOG	3400	1200	2200

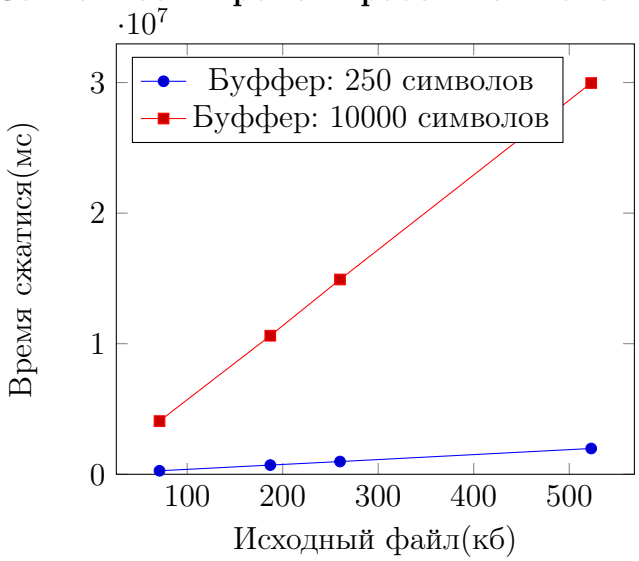
Здесь комбинация алгоритмов сработала намного лучше, чем только коды Хаффмана, так как в данных файлах много повторяющихся последовательностей символов.

Зависимость степени сжатия от максимальной размерности буффера LZ77:



Размер исходного файла(кб)	Буффер: 250 символов(кб)	Буффер: 10000 символов(кб)
71	63	28
187	168	75
260	230	101
523	468	206

Зависимость времени работы от максимальной размерности буффера LZ77:



Размер исходного файла(кб)	Буффер: 250 символов(мс)	Буффер: 10000 символов(мс)
71	269512	4074003
187	704904	10612509
260	975079	14917370
523	1974595	29965070

Как видно, размерность буффера сильно влияет на время работы. При уменьшении размерности с 10000 на 250, время уменьшается в 30 раз. Степень сжатия при большом буффере значительно лучше, но при этом скорость работы возврастает в десятки раз.

Выводы

Таким образом был реализован алгоритм кодирования LZ77 и коды Хаффмана. Преимуществами LZ77 являются простота реализации и высокая скорость декодирования. Уменьшение длины буффера значительно ускоряет сжатие, но уменьшает степень сжатия. Недостатком является то, что если в тексте находится уникальный текст или мало повторяющихся символов, то эффективность кодирования будет минимальной. Поэтому данный алгоритм эффективен на текстах с повторяющимися символами, если таких очень мало, то лучше использовать коды Хаффмана. Но при этом если необходимо закожировать тексты с повторяющимися словами(например файлы с кодом или лог файлы) то степень сжатия только кодами Хаффмана будет значительно уступать комбинации LZ77 + Huffman.