Курсовой проект по курсу дискретного анализа: Алгоритм LZ77

Выполнил студент группы М8О-308Б-20 МАИ *Попов Матвей*.

Условие

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода. Вам будут даны входные файлы двух типов.

1. Первый тип: текст состоит только из малых латинских букв. В ответ на него вам нужно вывести тройки, которыми будет закодирован данный текст.
2. Второй тип: вам даны тройки (<offset, len, char>) в которые был сжат текст из малых латинских букв, вам нужно его разжать.

Метод решения

Была реализована структура для хранения триплетов, а также функции *Compress*, которая принимает на вход строку и возвращает массив триплетов, и *Decompress*, которая принимает на вход массив триплетов и декодирует их в строку.

Описание программы

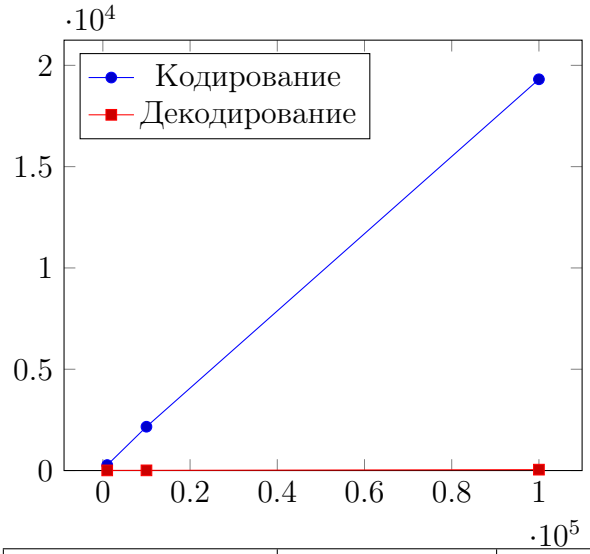
Программа состоит из пяти файлов: *Compress.hpp*, содержащий функцию для кодирования, *Decompress.hpp*, содержащий функцию для декодирования, *TTriplet.hpp*, содержащий структуру триплетов, *main.cpp* и *Makefile* для сборки программы.

Дневник отладки

1. Было принято решение сперва написать функцию для декодирования, чтобы в дальнейшем была возможность проверить корректность работы функции кодирования.
2. Функция для декодирования была написана без ошибок
3. В функции для кодирования была допущена ошибка при работе со скользящим окном

Тест производительности

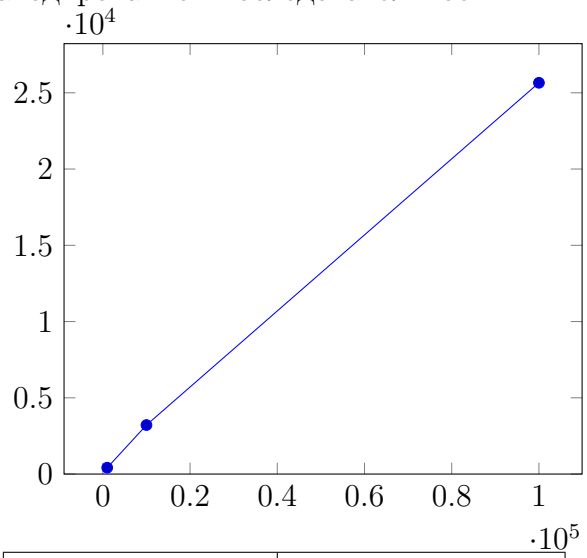
По оси X — количество символов в тексте, по оси Y — время работы алгоритмов в мс (меньше — лучше).



Кол-во символов	Кодирование	Декодирование
1000	271	7
10000	2163	9
100000	19312	44

Тест сжатия

По оси X — количество символов в исходной строке, по оси Y — количество триплетов в закодированной последовательности.



Кол-во символов	Кол-во триплетов
1000	416
10000	3214
100000	25656

Недочёты

Отсутствует буферизация закодированной части текста, то есть поиск наибольшей подстроки, которую предстоит закодировать, осуществляется во всей уже закодированной подстроке. Это обеспечивает наименьшую последовательность триплетов, но негативно сказывается на времени работы кодировки. Помимо этого алгоритм поиска наивный, из-за чего сложность по времени в худшем случае является $O(n^3)$, что хуже, если бы в основе поиска лежало бы суффиксное дерево ($O(n^2)$), однако тест производительности показывает, что даже с наивным алгоритмом сложность близка к линейной.

Выводы

Таким образом был реализован алгоритм кодирования LZ77. Его преимуществами являются простота реализации и высокая скорость декодирования. Недостатком является то, что алгоритм кодирования неэффективен на малых объёмах данных, ведь даже на тексте в 100000 символов итоговый размер триплетов превышает размер исходной строки, однако по графику видно, что с увеличением длины строки отношение количества триплетов к длине строки уменьшается, а значит алгоритм работает.