Курсовой проект по курсу дискретного анализа: LZ77

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

Вариант: Алгоритм LZ77

Реализуйте алгоритм LZ77. Без окна, поиск производится по всему просмотренному ранее тексту.
Формат ввода
Вам будут даны тесты двух типов.
Первый тип: compress <text>
Текст состоит только из малых латинских букв. В ответ на него вам нужно вывести тройки, которыми будет закодирован данный текст. Если последний символ в тройке не является буквой, не выводите его.
Второй тип: decompress <triplets>
Вам даны тройки (<offset, len, char>) в которые был сжат текст из малых латинских букв, вам нужно его разжать. В строке может отсутствовать char и это значит что далее букв нет (аналогично кодированию выше)
Формат вывода
В ответ на тест первого типа вам нужно вывести тройки, каждая в отдельной строке, которыми будет закодирован данный текст. На тест второго типа выведите разжатый текст.

Метод решения

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.

Описание программы

Была реализована структура для хранения Node, а также функции *code()*, которая принимает на вход строку и возвращает массив Node, и *decode()*, которая принима-

ет на вход массив Node и декодирует их в строку. И три вспомогательные функции: *matchFind(std::string buffer, std::string subString)* ищет максимальную совпадающую строку в буфере; *decode(codeFromInput())* считывает закодированный текст из файла в вектор Node; *textFromInput()* считывает из файла в string, добавляя междсу строками.

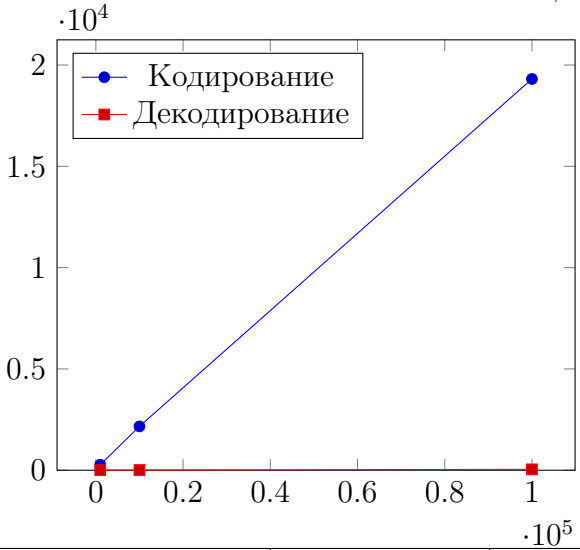
Дневник отладки

1. 6 дек 2023, 05:16:24 WA на 5 тесте.

   Причина: не использовал метод «просмотра в будущее», поэтому программа сжимала неправильно и неэффективно.

Тест производительности

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



Кол-во символов	Кодирование	Декодирование
1000	251	5
10000	2120	4
100000	19532	45

Выводы

Таким образом был реализован алгоритм кодирования LZ77. Его преимуществами являются простота реализации и высокая скорость декодирования. 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.