Алгоритм LZW (Lempel-Ziv-Welch)

Федотов И. Н.

ДВФУ

2022

Что такое алгоритм LZW?

LZW (Lempel-Ziv-Welch) — это алгоритм сжатия данных без потерь, который используется для сжатия текстовых файлов, изображений, звуковых файлов и других типов данных. Был разработан Абрахамом Лемпелем, Якобом Зивом и Терри Уэлчем в 1984 году.

Применение

LZW — это очень эффективный метод сжатия данных, который может быть использован для сокращения размера файлов и ускорения передачи данных.

Как работает?

LZW использует словарь для сжатия данных. Он начинает с заполненного словаря, содержащего все возможные символы и их соответствующие коды. Затем он проходит по исходным данным, считывая символы и сравнивая их с содержимым словаря.

Процесс сжатия

Когда LZW встречает последовательность символов, которая еще не встречалась, он добавляет ее в словарь и присваивает ей новый код. Затем он заменяет последовательность символов на новый код, который меньше по размеру, чем исходная последовательность символов. Если LZW встречает последовательность символов, которая уже есть в словаре, он заменяет ее на соответствующий код из словаря.

По шагам

1. Функция создает словарь.
2. Функция читает входные данные, символ за символом, и записывает их в буфер.
3. Функция начинает с чтения первого символа из буфера и добавляет его в текущую строку, начинает читать следующий символ и добавляет его в текущую строку, пока она не найдет такую последовательность символов, которой нет в словаре.
4. Функция находит последовательность символов, которой нет в словаре, она добавляет эту последовательность в словарь и назначает ей новый код.
5. Функция повторяет шаги 3-4 для оставшихся символов в буфере.
6. После обработки всех символов входных данных функция записывает оставшиеся коды символов в выходной поток.
7. Функция возвращает выходной поток, который содержит сжатые данные в формате последовательности кодов символов.

Пример сжатия

index	curr	isSeen?	encode	append
a	a	да	ничего	никакой
ab	ab	нет	0	ab / 4
abd	bd	нет	0, 1	bd / 5
abda	da	нет	0, 1, 3	da / 6
abdab	ab	да	без изменений	никакой
abdabc	abc	нет	0, 1, 3, 4	abc / 7

Процесс распаковки

Для распаковки сжатого файла используется обратный процесс. Коды заменяются на их соответствующие символы из словаря, который был использован для сжатия данных. Расшифровка данных происходит благодаря тому, что словарь сжатия и словарь распаковки одинаковы.

По шагам

1. Распаковка начинается с инициализации таблицы символами алфавита.
2. Функция читает закодированные данные, начиная с первого кода.
3. Функция использует текущий код для извлечения соответствующей последовательности символов из таблицы.
4. После извлечения последовательности символов, функция добавляет ее в выходные данные.
5. После добавления последовательности символов в выходные данные, функция обновляет таблицу
6. Функция продолжает чтение входных данных и извлечение последовательностей символов
7. По завершении функции, выходные данные содержат исходную последовательность символов, а таблица содержит все последовательности, которые были

Пример распаковки

index	dict	encode	curr	append
0	0 = a	a	никакая	никакая
0, 1	1 = b	ab	a	ab / 4
0, 1, 3	3 = d	abd	b	bd / 5
0, 1, 3, 4	4 = ab	abdab	d	da / 6
0, 1, 3, 4, 2	2 = c	abdabc	ab	abc / 7
0, 1, 3, 4, 2, 2	2 = c	abdabcc	c	cc / 8

Оценка производительности

Время работы алгоритма LZW может быть оценено как $O(n \log k)$, где $\log k$ - это высота бинарного дерева, используемого для хранения словаря. Эта оценка производительности объясняется тем, что время работы алгоритма зависит от количества символов, которые нужно закодировать, и количества символов в алфавите. Кроме того, дополнительное время затрачивается на построение словаря и кодирование символов.

Оценка по памяти

Оценка по памяти для алгоритма LZW может быть выражена как $O(n + k)$, где первый член означает, что необходимо хранить входные данные, а второй член означает, что необходимо хранить словарь, который может содержать до k элементов, что может быть большим для больших алфавитов, но в общем случае оценка по памяти алгоритма LZW является линейной по размеру входных данных.

Спасибо за внимание!