ПОДСТАНОВОЧНЫЕ ИЛИ СЛОВАРНО-ОРИЕНТИРОВАННЫЕ АЛГОРИТМЫ СЖАТИЯ ИНФОРМАЦИИ. МЕТОДЫ ЛЕМПЕЛЯ-ЗИВА.

- LZ77
- LZSS
- LZ78
- LZW

СЛОВАРНЫЕ АЛГОРИТМЫ

- менее математически обоснованы
- имеют более практический характер

LZ77

- 1977г.
- Абрахам Лемпель (Abraham **L**empel) и Якоб Зив (Jacob **Z**iv)
- простота
- высокая эффективность сжатия

ОСНОВНАЯ ИДЕЯ LZ77

второе и последующие вхождения некоторой строки символов в сообщении заменяются **ссылками** на ее первое вхождение

ПРИНЦИП «СКОЛЬЗЯЩЕГО ОКНА»

- разделено на две неравные части
- **первая**, большая по размеру, включает уже просмотренную часть сообщения **(словарь – несколько Кб)**
- **вторая**, намного меньшая, является **буфером (<=100байт)**, содержащим еще незакодированные символы входного потока

ЧТО ДЕЛАЕТ АЛГОРИТМ LZ77

- пытается найти в **словаре** фрагмент, совпадающий с содержимым **буфера**
- выдает коды, состоящие из трех элементов:

  1. **смещение** в словаре относительно его начала подстроки, совпадающей с началом содержимого буфера;
  2. **длина** этой **подстроки**;
  3. **первый символ буфера**, следующий за подстрокой;

ПРИМЕР

Закодировать строку «**КРАСНАЯ_КРАСКА**»

СЛОВАРЬ (8)	БУФЕР (5)	КОД
********	КРАСН	<0,0,’К’>
*******К	РАСНА	<0,0,’Р’>
******КР	АСНАЯ	<0,0,’А’>
*****КРА	СНАЯ_	<0,0,’С’>
****КРАС	НАЯ_К	<0,0,’Н’>
***КРАСН	АЯ_КР	<5,1,’Я’>
*КРАСНАЯ	_КРАС	<0,0,’_’>
КРАСНАЯ_	КРАСК	<0,4,’К’>
АЯ_КРАСК	А****	<0,0,’А’>

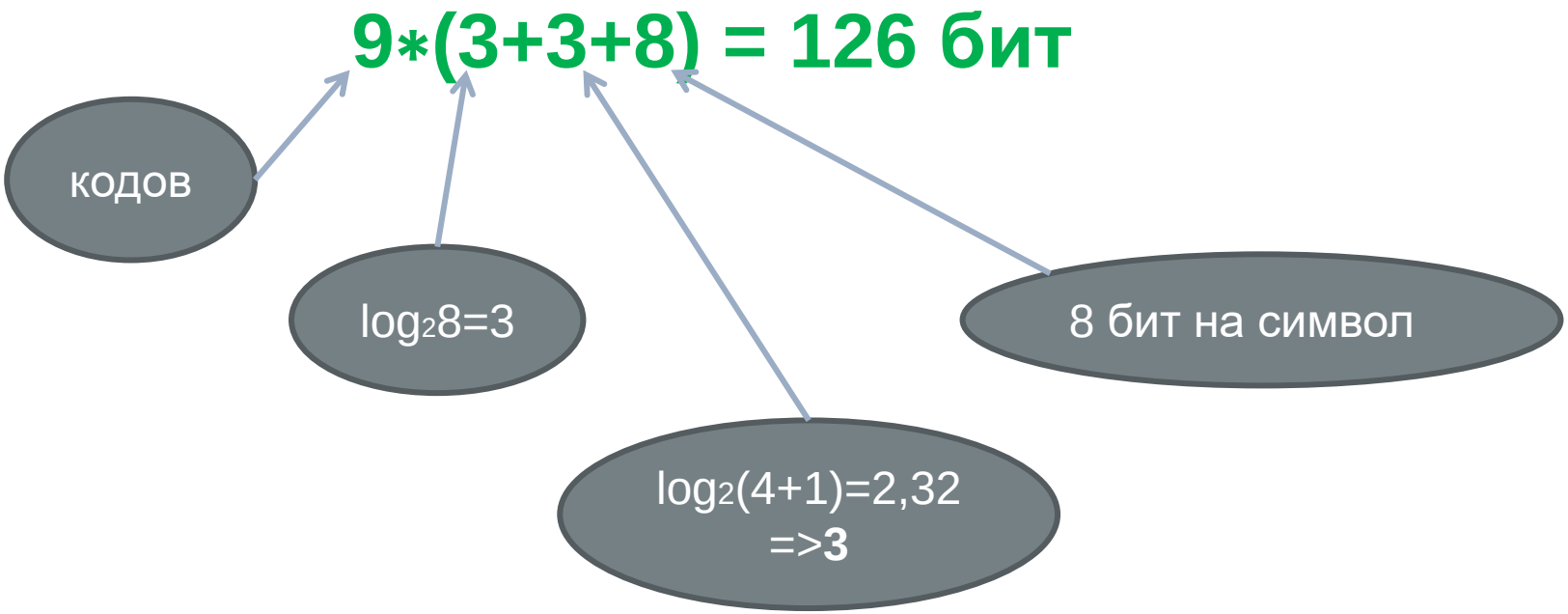
- в последней строчке буква «А» берется не из словаря, т.к. она последняя

ПОДСЧЕТ ДЛИНЫ КОДА

- длина подстроки не может быть больше размера буфера
- смещение не может быть больше размера словаря **-1**
- длина двоичного кода смещения будет округленным в большую сторону $\log_2$ (*размер словаря*)
- длина двоичного кода для длины подстроки будет округленным в большую сторону $\log_2$ (*размер буфера+1*)
- символ кодируется **8 битами** (например, ASCII+)

ПОДСЧЕТ ДЛИНЫ КОДА

длина полученного кода для **КРАСНАЯ_КРАСКА** равна



LZSS

- 1982 г.
- Сторер (Storer) и Шиманский (Szimanski)
- на базе LZ77
- отличается от LZ77 производимыми кодами

ОСНОВНАЯ ИДЕЯ LZSS

- выдаваемый код начинается с **однобитного префикса**, различающего собственно код от незакодированного символа
- **код состоит из пары:**

  смещение и **длина** (такими же как и для LZ77)

ЧТО ДЕЛАЕТ АЛГОРИТМ LZSS

- окно сдвигается ровно на длину найденной подстроки **или на 1**, если не найдено вхождение подстроки из буфера в словарь
- длина подстроки в LZSS всегда **больше 0**, поэтому длина двоичного кода для длины подстроки — это округленный до большего целого **$\log_2$** от длины **буфера**

ПРИМЕР

Закодировать строку «**КРАСНАЯ_КРАСКА**»

СЛОВАРЬ (8)	БУФЕР (5)	КОД	ДЛИНА КОДА
********	КРАСН	0,’К’	9
*******К	РАСНА	0,’Р’	9
******КР	АСНАЯ	0,’А’	9
*****КРА	СНАЯ_	0,’С’	9
****КРАС	НАЯ_К	0,’Н’	9
***КРАСН	АЯ_КР	1<5,1>	7
**КРАСНА	Я_КРА	0,’Я’>	9
*КРАСНАЯ	_КРАС	0,’_’	9
КРАСНАЯ_	КРАСК	1<0,4>	7
НАЯ_КРАС	КА***	1<4,1>	7
АЯ_КРАСК	А****	1<0,1>	7

ПОДСЧЕТ ДЛИНЫ КОДА

длина полученного кода для **КРАСНАЯ_КРАСКА** равна

7*9+4*7 = 91 бит

НЕДОСТАТКИ LZ77 И LZSS

- невозможность кодирования подстрок, отстоящих друг от друга на расстоянии, большем длины словаря
- длина подстроки, которую можно закодировать, ограничена размером буфера

LZ78

- 1978 г.
- Абрахам Лемпель (Abraham **L**empel) и Якоб Зив (Jacob **Z**iv)
- на базе LZ77
- лишен недостатков LZ77

ОСНОВНАЯ ИДЕЯ LZ78

- не использует “скользящее” окно
- хранит словарь из уже просмотренных фраз

ЧТО ДЕЛАЕТ АЛГОРИТМ LZ78

1. при старте словарь содержит только одну пустую строку (**строку длины 0**)
2. алгоритм считывает символы сообщения до тех пор, пока накапливаемая подстрока входит целиком в одну из фраз словаря
3. как только эта строка перестанет соответствовать хотя бы одной фразе словаря, алгоритм генерирует код, состоящий из индекса строки в словаре, которая до последнего введенного символа содержала входную строку, и символа, нарушившего совпадение
4. затем в словарь добавляется введенная подстрока

если словарь уже заполнен, то из него предварительно удаляют фразу, используемую менее всех в сравнениях

ПРИМЕР

Закодировать строку «**КРАСНАЯ_КРАСКА**», используя словарь длиной **16 фраз**

ВХОДНАЯ ФРАЗА (В СЛОВАРЬ)	КОД	ПОЗИЦИЯ СЛОВАРЯ
“”		0
К	<0,К>	1
Р	<0,Р>	2
А	<0,А>	3
С	<0,С>	4
Н	<0,Н>	5
АЯ	<3,Я>	6
“_”	<0, ‘_’>	7
КР	<1,Р>	8
АС	<3,С>	9
КА	<1,А>	10

ПОДСЧЕТ ДЛИНЫ КОДА

- **указатель** на любую фразу такого словаря — это число от 0 до 15, для его кодирования достаточно **четырех бит** (1610=11112)
- ключевым для размера получаемых кодов является размер словаря во фразах, т.к. **каждый код** при кодировании по методу LZ78 **содержит номер фразы в словаре**
- коды имеют постоянную длину, равную округленному в большую сторону **log2** размера словаря +8 (это количество бит в байткоде расширенного ASCII)

ПОДСЧЕТ ДЛИНЫ КОДА

длина полученного кода для **КРАСНАЯ_КРАСКА** равна

10*(4+8) = 120 бит

LZW

- 1984 г.
- Уэлч (Welch)
- модификация LZ78

ЧТО ДЕЛАЕТ АЛГОРИТМ LZW

Шаг 1. Инициализация словаря всеми возможными односимвольными фразами (обычно 256 символами расширенного ASCII). Инициализация входной фразы **w** первым символом сообщения.
Шаг 2. Считать очередной символ **K** из кодируемого сообщения.
Шаг 3. Если **КОНЕЦ СООБЩЕНИЯ**

Выдать код для **w**
Конец

Если фраза **wK** уже есть в словаре
Присвоить входной фразе значение **wK**
Перейти к Шагу 2

Иначе
Выдать код **w**
Добавить **wK** в словарь
Присвоить входной фразе значение **K**
Перейти к Шагу 2

ПРИМЕР

Закодировать строку «**КРАСНАЯ_КРАСКА**», размер словаря **500 фраз**

ВХОДНАЯ ФРАЗА, wK (В СЛОВАРЬ)	КОД для w	ПОЗИЦИЯ СЛОВАРЯ
ASCII++		0-255
КР	0’К’	256
РА	0’Р’	257
АС	0’А’	258
СН	0’С’	259
НА	0’Н’	260
АЯ	0’А’	261
Я_	0’Я’	262
_К	0’_’	263
КРА	<256>	264
АСК	<258>	265
КА	0’К’	266
А	0’А’	267

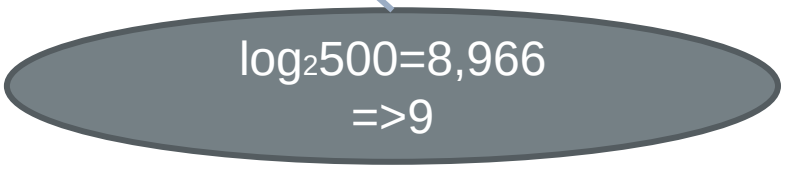
ПОДСЧЕТ ДЛИНЫ КОДА

- как и в случае с LZ78 для LZW ключевым для размера получаемых кодов является размер словаря во фразах
- **LZW-коды** имеют **постоянную длину**, равную округленному в большую сторону **$\log_2$** размера словаря

ПОДСЧЕТ ДЛИНЫ КОДА

длина полученного кода для **КРАСНАЯ_КРАСКА** равна

12*9 = 108 бит



ВАЖНО!

при переполнении словаря, т. е. когда необходимо внести новую фразу в полностью заполненный словарь, из него удаляют либо наиболее редко используемую фразу, либо все фразы, отличающиеся от одиночного символа

ИТОГ

- несмотря на свою простоту и, казалось бы, не слишком уж большую эффективность, эти алгоритмы до сих пор широко применяются в разнообразных областях IT-сферы
- их плюс — **простота и быстродействие**, а на их принципах и их комбинациях могут быть основаны более сложные и эффективные алгоритмы

ПРАВА НА ИСПОЛЬЗОВАНИЕ

- LZ77, LZ78 и LZSS разработаны математиками и могут использоваться свободно!
- LZW является запатентованным и, таким образом, представляет собой интеллектуальную собственность. Его безлицензионное использование, особенно на аппаратном уровне может повлечь за собой неприятности

УПРАЖНЕНИЕ

Закодировать сообщение: **AABCDAACCCCDBB** и вычислить длины (в битах) полученных кодов, используя алгоритмы

LZ77 (словарь – 12 символов, буфер – 4)

LZSS (словарь – 12 символов, буфер – 4)

LZ78 (словарь – 16 фраз)

LZW (словарь – 500 фраз)