Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ*

## *НА ТЕМУ:*

## *«Методы сжатия изображений»*

| | | | |
|---|---|---|---|
| Студент | ИУ7-54Б | | Ланкин Д. Л. |
| | (Группа) | (Подпись, дата) | (И. О. Фамилия) |
| Руководитель НИР | | | Кострицкий А. С. |
| | | (Подпись, дата) | (И. О. Фамилия) |
| Консультант | | | в |
| | | (Подпись, дата) | (И. О. Фамилия) |

*2023 г.*

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Проблема сжатия данных является актуальной уже на протяжении длительного времени. С постоянным ростом объема данных, эффективное сжатие становится необходимостью для экономии пространства хранения и более быстрой передачи информации. Вопрос сжатия изображений является одним из актуальных.

Сжатие изображений является процессом уменьшения размера изображения. Оно позволяет сократить объем данных, улучшить скорость передачи и сэкономить место на устройствах хранения. Общим свойством алгоритмов сжатия можно считать то, что они выполняются над всем изображением.[1]

Таким образом, сжатие изображений является важным аспектом обработки данных, который позволяет сократить объем информации и улучшить скорость передачи.

Целью данной работы является изучение и описание известных алгоритмов сжатия изображений.

Для достижения поставленной цели необходимо выполнить следующие задачи:

— изучить принципы сжатия данных и понять, как они применяется к изображениям;

— классифицировать методы сжатия изображений;

— описать рассматриваемые алгоритмы на формальном языке;

— выделить преимущества и недостатки рассматриваемых алгоритмов;

— сравнить рассматриваемые алгоритмы.

# 1 Аналитический раздел

Сжатие иногда называют исходным кодированием, поскольку оно пытается удалить избыточность в строке из-за предсказуемости источника. Для конкретной изображения коэффициент сжатия - это отношение размера сжатого выхода к исходному размеру изображения. [2]

Когда человечество столкнулось с необходимостью сжатия, начали появляться первые алгоритмы. Они реализовывали сжатие без потери какой-либо информации. Однако в последнее время стали появляться новые классы изображений, сжатие над которыми ранее существовавшими алгоритмами стало неэффективным – они почти не сжимались, хотя обладали явной избыточностью. [3]

Тогда возникла необходимость использования алгоритмов с потерями.

Алгоритмы сжатия можно разделить на 2 главных класса: без потерь (англ. lossless) и с потерями (англ. lossy). Данные, сжатые с использованием первого подхода могут быть восстановлены точь-в-точь, когда сжатие с потерями подразумевает потерю некоторых данных. [4]

**TODO: классифицировать алгоритмы сжатия.**

Процесс сжатия данных состоит из 2 этапов:

— моделирование;

— кодирование.

## 1.1 Моделирование

На этом этапе происходит разбиение входных данных по частоте появления. Наиболее частые последовательности битов обладают большим значением вероятности и наоборот. Таким образом, на данном этапе происходит присвоение вероятностей для последовательностей битов.

## 1.2 Кодирование

Зная длину кода, соответствующую данной последовательности в зависимости от ее частоты (1.1), необходимо составить уникальную цепочку символов, которая будет единым образом сопоставлять исходную последовательность с кодированной.

**TODO:**

## 1.3 Связь моделирования и кодирования

Отношение между этими этапами было установлено в теореме Шеннона: символ, вероятность появления которого равна $p$ будет представлен $-\log p$ битами. В таком случае, высокочастотный символ будет закодирован минимальным количеством битов, а низкочастотный — максимальным. Тогда длину последовательности символов можно определить с помощью формулы

$$-\sum_{i=1}^{N} p_i \log p, \tag{1.1}$$

где $N$ — количество последовательностей, $p_i$ — вероятность $i$-й последовательности.

Эта величина называется *энтропией* вероятностного распределения.

## 1.4 LZSS

LZSS, описанный Сторером (англ. Storer) и Шимански (англ. Szymanski"), является предком алгоритма LZ77 (Lempel-Ziv-77). Он использует метод динамического словаря. Метод синтаксического анализа, используемый LZSS, является жадным (без предварительного просмотра).

Словарь на каждом шаге состоит из всех односимвольных подстрок и всех подстрок взодных данных, которые начинаются с уже сжатого префикса входных данных и заканчиваются перед концом анализируемой фразы. Каждое кодовое слово состоит из битового флага, который обозначает, относится ли кодовое слово к типу односимвольных подстрок или всех подстрок. Кодовые слова первого типа — односимвольных фраз — просто состоят из соответствующего символа. Кодовые слова второго типа состоят из двух элементов: указателя на начальное местоположение последнего вхождения фразы и длины фразы. Как указатель, так и информация о длине могут быть представлены фиксированным числом битов. [4]

**Применение.** Утилита *gzip*, распространяемая Free Software Foundation, использует модификацию алгоритма LZSS.

## 1.5 LZW

LZW использует жадный синтаксический анализ как и LZ77. Разница заключается в том, что изначально словарь состоит только из односимвольных подстрок и после каждого шага синтаксического анализа проанализированная фраза объединяется с первым символом несжатой части входных данных и вставляется в словарь как новая фраза. Каждой такой фразе присваивается $|D| + 1$ в качестве кодового слова, где $|D|$ — количество фраз в словаре. В таком случае на любом шаге каждая фраза представлена $\log_2 |D|$ битами.

**Применение.** Алгоритм LZW с небольшими изменениями в обслуживании словаря используется в GIF.

На листингах А.1–А.2 представлен псевдокод алгоритма LZW.

## 1.6 Проблема переполнения динамического словаря

Важной проблемой при реализации методов Lempel — Ziv, основанных на работе с динамическим словарем, является ограничение памяти в словаре. Есть несколько способов борьбы с этим явлением.

1. **Ограничение размера.** При достижении максимальной вместимости происходит пересоздание словаря.

2. **Замораживание добавления.** При этом подходе в момент достижения ограничения приостанавливается вставка новых фраз, а дальнейшее сжатие реализуется без каких-либо изменений.

3. **Удаление наименее недавно использованных (англ. least recently used — LRU).** В данном случае происходит удаление пары фраза/кодовое слово из словаря при условии, что она использовалась наиболее давно из всего набора. Ключевое слово может быть переиспользовано для новой фразы, вставляемой далее. Этот подход требует хранения времени последнего использования каждой фразы. В этом заключается главный минус подхода.

4. **Удаление наименее часто попадающихся фраз/кодовых слов (англ. least frequently used — LFU).** Частота фразы определяется как количество обращений к ней, нормализованное относительно количества

обращений ко всему словарю с момента вставки этой фразы/кодового слова. Такой подход требует обновления информации о частоте каждой фразы на каждом шаге.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. *Белов А. С.* Адаптивный способ сжатия изображений // T-Comm. №9. — 2013. — URL: https://cyberleninka.ru/article/n/adaptivnyy-sposob-szhatiya-izobrazheniy (дата обращения: 29.10.2023).
2. *Bell T.*, *Witten I.*, *Cleary J.* Modeling for Text Compression. // ACM Comput. Surv. — 1989. — Дек. — Т. 21. — С. 557—591. — DOI: 10.1145/76894.76896.
3. *Петрянин Д. Л.*, *Ассонова М. Л.*, *Белов А. Г.* Оценка проблем при архивации изображений с потерями // Новые информационные технологии в автоматизированных системах. №917. — 2014. — URL: https://cyberleninka.ru/article/n/otsenka-problem-pri-arhivatsii-izobrazheniy-s-poteryami (дата обращения: 29.10.2023).
4. *Sayood K.* Lossless Compression Handbook. — Elsevier Science, 2002. — (Communications, Networking and Multimedia). — ISBN 9780080510491. — URL: https://books.google.ru/books?id=LjQiGwyabVwC.

# ПРИЛОЖЕНИЕ А

Листинг А.1 – Кодирование методом LZW

```
function LZWEncoding(s1)
    table = create new unordered_map<string, int>

    for i from 0 to 255 do
        ch = convert i to character
        table[ch] = i
    end for

    p = ""
    c = ""
    p += s1[0]
    code = 256
    output_code = create new empty vector of integers

    for i from 0 to length of s1 - 1 do
        c += s1[i + 1] if i != length of s1 - 1

        if (p + c) is in table then
            p = p + c
        else
            append table[p] to output_code
            table[p + c] = code
            increment code by 1
            p = c
        end if

        c = ""
    end for

    print p, table[p]
    append table[p] to output_code

    return output_code
end function
```

Листинг А.2 – Кодирование методом LZW

```
function LZWDecode(op)
```

```
    table = create new empty map from integer to string
    decodedString = ""

    for i from 0 to 255 do
        ch = convert i to character
        table[i] = ch
    end for

    old = op[0]
    n = 0
    s = table[old]
    c = s[0]

    count = 256

    for i from 0 to length of op - 2 do
        n = op[i+1]

        if table does not contain n then
            s = table[old] + c
        else
            s = table[n]
        end if

        append s to decodedString

        c = s[0]
        table[count] = table[old] + c
        increment count by 1
        old = n
    end for

    return decodedString
end function
```