ДОДАТОК А

Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії

1. Software agents for learning resources of digital library, DA Milashenko, Sergiy D Makovetskiy, Natalya S Lesna, Gesellschaft für Informatik eV. – 2003.

2. Hennadii Falatiuk. Investigation of Architecture and Technology Stack for e-Archive System / Hennadii Falatiuk, Mariya Shirokopetleva, Zoia Dudar, IEEE – 2019, DOI: 10.1109/PICST47496.2019.9061407

ДОДАТОК Б

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

UNICHECK
by Turnitin

Ім'я користувача:
Нечволод Вадим Юрійович каф. ПІ

ID перевірки:
1016348595

Дата перевірки:
11.06.2024 18:40:15 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
11.06.2024 20:20:36 EEST

ID користувача:
94949

Назва документа: 2024_М_ПІ_ІПЗм_22_1_Жихарський_П_О_скорочений

Кількість сторінок: 47 Кількість слів: 8810 Кількість символів: 63380 Розмір файлу: 1.44 MB ID файлу: 1016151867

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

7.17%
Схожість

Найбільша схожість: 1.4% з Інтернет-джерелом (http://ni.biz.ua/4/4_13/4_130408_arhivatsiya-informatsii-i-metodi-rabot...

7.03% Джерела з Інтернету 156 ..... Сторінка 49

0.52% Джерела з Бібліотеки 11 ..... Сторінка 50

0.33% Цитат

Цитати 4 ..... Сторінка 51

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 8 сторінок

ДОДАТОК В

Приклади коду

```
public static class Program
{
    public static async Task<int> Main(string[] args)
    {
        var registrations = new ServiceCollection();
        registrations.AddTransient<ICompressAlgorithm, Adaptive>();
        registrations.AddTransient<ICompressAlgorithm, Huffman>();
        registrations.AddTransient<ICompressAlgorithm, LZ77>();
        registrations.AddTransient<ICompressAlgorithm, RLE>();
        registrations.AddTransient<ICompressAlgorithm,
ArithmeticCoding>();

        var registrar = new TypeRegistrar(registrations);
        var app = new CommandApp(registrar);
        app.Configure(config =>
        {
            config.AddCommand<CompressCommand>("compress");
            config.AddCommand<DecompressCommand>("decompress");
        });

        return await app.RunAsync(args);
    }
}

public class CompressCommand : Command<CompressCommandSettings>
{
    private readonly IEnumerable<ICompressAlgorithm> compressAlgorithms;

    public CompressCommand(IEnumerable<ICompressAlgorithm>
compressAlgorithms)
    {
        this.compressAlgorithms = compressAlgorithms;
    }

    public override int Execute(CommandContext context,
CompressCommandSettings settings)
    {
        var compressAlgorithm = GetCompressAlgorithm(settings.Algorithm);
        var sourceFileString = File.ReadAllText(settings.SourceFilePath,
Encoding.UTF8);
        var timer = new Stopwatch();

        timer.Start();
        compressAlgorithm.Encode(sourceFileString, settings.EndFilePath);
        timer.Stop();

        var originalSize = new FileInfo(settings.SourceFilePath).Length;
        var compressedSize = new FileInfo(settings.EndFilePath).Length;
        var compressRation = (double) compressedSize / originalSize;

        DisplayResultTable(settings.Algorithm, originalSize,
compressedSize, compressRation, timer.ElapsedMilliseconds);
```

```
        return 0;
    }

    private static void DisplayResultTable(string algorithm, long
originalSize, long compressedSize, double compressionRatio, long time)
    {
        var table = new Table();

        table.AddColumn("Algorithm");
        table.AddColumn("Original size");
        table.AddColumn("Compressed size");
        table.AddColumn("Compression ratio");
        table.AddColumn("Time");

        table.AddRow(algorithm, originalSize + "b", compressedSize + "b",
compressionRatio.ToString(CultureInfo.CurrentCulture), time.ToString() +
"ms");

        AnsiConsole.Write(table);
    }

    private ICompressAlgorithm GetCompressAlgorithm(string algorithmType)
    {
        switch (algorithmType)
        {
            case "A" : return compressAlgorithms.First(x => x.Type ==
CompressAlgorithm.ArithmeticCoding);
            case "H" : return compressAlgorithms.First(x => x.Type ==
CompressAlgorithm.Huffman);
            case "L" : return compressAlgorithms.First(x => x.Type ==
CompressAlgorithm.LZ77);
            case "R" : return compressAlgorithms.First(x => x.Type ==
CompressAlgorithm.RLE);
            case "M" : return compressAlgorithms.First(x => x.Type ==
CompressAlgorithm.Adaptive);
        }

        return null;
    }
}
```

ДОДАТОК Г

Слайди презентації

Дослідження методів та алгоритмів компресії даних

Жихарський Петро Олександрович, ІПЗм-22-1
Науковий керівник: доц. Каук Віктор Іванович

17 червня 2024

Актуальность дослідження

- Актуальність роботи з дослідження методів та алгоритмів компресії даних зумовлена обсягом генерованих і збережених даних. Мультимедійні дані та інші джерела вимагають ефективних методів зберігання і передачі інформації.
- Пристрої з обмеженими обчислювальними ресурсами потребують методів, що дають змогу мінімізувати використання пам'яті та пропускної здатності мережі.

Постановка задачі

- проаналізувати та обрати алгоритми для компресії даних;
- визначити метрики, які будуть використані для проведення експерименту та подальшого оцінювання;
- визначити функціональні вимоги для консольного застосунку, який буде використано для проведення експерименту, та розробити його;
- виміряти та підрахувати значення обраних метрик для кожного з алгоритмів; надати рекомендації щодо використання кожного з методів.

Вибір методів та алгоритмів для дослідження

- RLE
- Huffman
- LZ77
- Арифметичне кодування
- Адаптивний алгоритм

Планування експериментального дослідження

Для проведення дослідження були обрані наступні метрики:

- Початковий розмір файлу
- Розмір файлу після компресії
- Коефіцієнт компресії
- Час компресії
- Час декомпресії

Планування експериментального дослідження

Для проведення дослідження були обрані наступні типи файлів:

- Документ із текстом у форматі txt;
- Документ із текстом у форматі docx;
- Зображення у форматі bmp;
- Зображення у форматі jpg.

Опис програмного забезпечення, що було використано у дослідженні

- Проект для проведення дослідження – консольний застосунок на платформі .NET 8
- Бібліотека для створення CLI інтерфейсу – Spectre.Console
- Мова програмування – C# 11
- Усі алгоритми були реалізовані власноруч без застосування сторонніх бібліотек

```
> ./CompressApp.Console
USAGE:
    CompressApp.Console.dll [OPTIONS] <COMMAND>

OPTIONS:
    -h, --help       Prints help information
    -v, --version    Prints version information

COMMANDS:
    compress      Compress given file
    decompress    Decompress given file
```

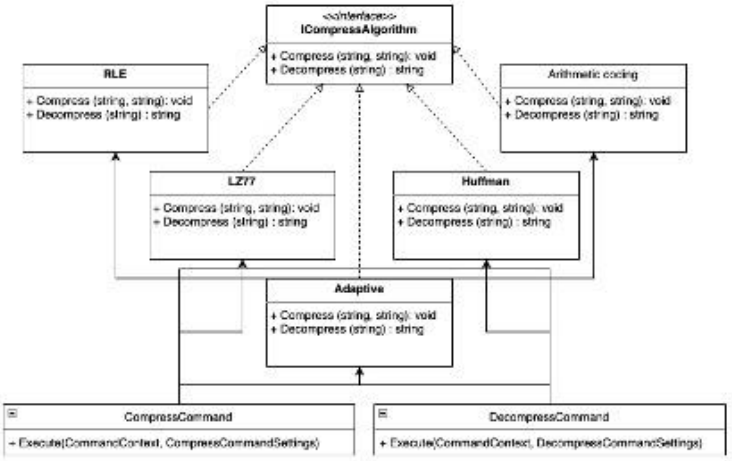
```
> ./CompressApp.Console compress -h
DESCRIPTION:
Compress given file

USAGE:
    CompressApp.Console.dll compress [OPTIONS]

OPTIONS:
    -h, --help       Prints help information
    -v, --version    Prints version information
    -s               Show statistic data
    -p               Path of source file
    -P               Path of end file
    -a               Algorithm for compression (A - arithmetic coding, H -
                     huffman, L - LZ77, R - RLE, M - adaptive)
```

7

Опис програмного забезпечення, що було використано у дослідженні



8

Результати експерименту

Алгоритм Хаффмана

- Висока швидкість роботи
- Високий коефіцієнт компресії
- Великий час декомпресії у порівнянні із часом компресії

Назва файлу	Початковий розмір	Розмір після компресії	Коефіцієнт компресії	Час компресії	Час декомпресії
1.txt	663442b	217140b	0,327	27ms	1989ms
2.txt	1105612b	361310b	0,326	42ms	5076ms
3.txt	4133129b	1371817b	0,331	156ms	19914ms
1.doc	216144b	196021b	0,906	189ms	21270ms
2.doc	372732b	310022b	0,831	290ms	36123ms
3.doc	1402567b	1066562b	0,760	1394ms	178884ms
1.bmp	818058b	729979b	0,892	902ms	101754ms
2.bmp	3275658b	2721545b	0,830	4898ms	280431ms
3.bmp	7372938b	5946655b	0,806	13933ms	483275ms
1.jpg	474441b	380324b	0,801	328ms	29185ms
2.jpg	744692b	577176b	0,775	552ms	43954ms
3.jpg	1372378b	1029390b	0,750	1153ms	104584ms

Результати експерименту

Алгоритм LZ77

- Висока швидкість роботи
- Задовільний коефіцієнт компресії лише на текстових файлах
- Висока швидкість декомпресії

Назва файлу	Початковий розмір	Розмір після компресії	Коефіцієнт компресії	Час компресії	Час декомпресії
1.txt	663442b	463119b	0,698	161ms	21ms
2.txt	1105612b	808896b	0,731	252ms	32ms
3.txt	4133129b	3034398b	0,734	1050ms	121ms
1.doc	216144b	303718b	1,405	124ms	17ms
2.doc	372732b	528100b	1,416	220ms	28ms
3.doc	1402567b	1993783b	1,421	802ms	95ms
1.bmp	818058b	1126186b	1,376	419ms	56ms
2.bmp	3275658b	4354731b	1,329	1422ms	182ms
3.bmp	7372938b	9694066b	1,314	3293ms	390ms
1.jpg	474441b	665728b	1,403	267ms	36ms
2.jpg	744692b	1037663b	1,393	366ms	47ms
3.jpg	1372378b	1905057b	1,388	688ms	82ms

Результати експерименту

Алгоритм RLE

- Висока швидкість роботи
- Коефіцієнт компресії завжди більше 1
- Висока швидкість декомпресії

Назва файлу	Початковий розмір	Розмір після компресії	Коефіцієнт компресії	Час компресії	Час декомпресії
1.txt	663442b	1092814b	1,647	19ms	32ms
2.txt	1105612b	1809457b	1,636	44ms	52ms
3.txt	4133129b	6807283b	1,647	138ms	217ms
1.doc	216144b	500263b	2,314	12ms	12ms
2.doc	372732b	865753b	2,322	23ms	24ms
3.doc	1402567b	3278302b	2,337	65ms	92ms
1.bmp	818058b	1890505b	2,31	37ms	52ms
2.bmp	3275658b	7567690b	2,31	159ms	261ms
3.bmp	7372938b	16966333b	2,3	314ms	585ms
1.jpg	474441b	1124113b	2,369	22ms	37ms
2.jpg	744692b	1755625b	2,357	42ms	46ms
3.jpg	1372378b	3218170b	2,344	64ms	91ms

Результати експерименту

Арифметичне кодування

- Дуже мала швидкість компресії
- Коефіцієнт компресії завжди більше 1

Назва файлу	Початковий розмір	Розмір після стискання	Коефіцієнт компресії	Час компресії
1.txt	663442b	1047574b	1,579	600min

Результати експерименту

Адаптивний алгоритм

- Висока швидкість роботи
- Задовільний коефіцієнт компресії лише на текстових файлах
- Висока швидкість декомпресії

Назва файлу	Початковий розмір	Розмір після компресії	Коефіцієнт компресії	Час компресії	Час декомпресії
1.txt	663442b	431532b	0,65	206ms	130ms
2.txt	1105612b	724185b	0,655	349ms	189ms
3.txt	4133129b	2734833b	0,66	1187ms	607ms
1.doc	216144b	399907b	1,85	239ms	151ms
2.doc	372732b	684951b	1,837	382ms	207ms
3.doc	1402567b	2578841b	1,836	1332ms	840ms
1.bmp	818058b	1388699b	1,697	754ms	409ms
2.bmp	3275658b	5597928b	1,7	2823ms	1445ms
3.bmp	7372938b	12851990b	1,743	6983ms	4404ms
1.jpg	474441b	848591b	1,788	457ms	248ms
2.jpg	744692b	1323973b	1,777	696ms	439ms
3.jpg	1372378b	2420543b	1,763	1231ms	667ms

13

Аналіз отриманих результатів

- Оптимальний методом компресії даних є алгоритм Хаффмана, з точки зору універсальності та коефіцієнту компресії. В той же час він має один суттєвий недолік, це швидкість декомпресії. Вона є дуже низька у порівняні із швидкістю компресії.
- Алгоритм LZ77, який має трохи меншу швидкість роботи ніж алгоритм Хаффмана та значно високу швидкість декомпресії, але має певну кількість недоліків, які не дають йому змогу конкурувати із алгоритмом Хаффмана. Перший і основний недолік це ефективність на лише на текстових файлах, інші типи файлів показали коефіцієнт компресії більше 1. Другий недолік це невисокий коефіцієнт компресії для текстових файлів у порівнянні із алгоритмом Хаффмана.

14

Публікація результатів

Висновки

- проаналізовано та обрано алгоритми для компресії даних
- розроблені бібліотеки із алгоритмами
- виміряно та підраховано значення обраних метрик для кожного з методів компресії даних за допомогою розробленого застосунку
- порівняно та проаналізовано отриманні дані
- За результатами дослідження опубліковано тези «Data Compression Algorithms» на 26-тій Міжнародній науковій та технічній конференції «Theoretical And Practical Aspects Of Modern Research»

ДОДАТОК Д

Апробація результатів роботи

ISU
INTERNATIONAL SCIENTIFIC UNITY

CERTIFICATE
of conference participant

it is hereby certified, that
PETRO ZHYKHARSKYI
took part in the XXVI International Scientific and Practical Conference
«THEORETICAL AND PRACTICAL ASPECTS OF MODERN RESEARCH»

June 5-7, 2024, Ottawa, Canada
12 Hours of Participation
(0,4 ECTS credits)

Head of the
organizing committee

Viktoriia Tsiundyk

Conference name -«XXVI Міжнародна науково-практична конференція «Theoretical and practical aspects of modern research»
Section name – Information technology and cyber security

DATA COMPRESSION ALGORITHMS

Petro Zhykharskyi
Master's degree student
Department of Software engineering
Kharkiv National University of Radioelectronics, Ukraine
petro.zhykharskyi@nure.ua

The amount of data is increasing rapidly, and more and more information is generated and processed every year. In this regard, it becomes important to be able to manage data effectively, not only to preserve information, but also to optimize its use and transfer. One of the key tools for this is data compression.

Data compression is the process of reducing the amount of data with minimal or no loss of information. Various compression algorithms and methods can significantly reduce the size of the data, saving their basic content and key characteristics. This is useful in various situations: saving storage space, speeding up data transmission, reducing the cost of Internet traffic and increasing the efficiency of information processing and analysis systems.

The main idea of data compression algorithms is to reduce the amount of data while saving its information value. Data compression is widely used in various fields such as compression of audio and video files, images, file archiving, and more.

Compression of audio and video allows you to significantly reduce the size of files required for their storage and online transmission. This is particularly important for streaming services and messengers, where high quality playback is required at minimal bandwidth costs.

Image compression is a key aspect in the visual content, where download speed and display graphic elements on different devices is important. Using compression algorithms allow to optimized images for different operating systems and screens, minimizing file size without loosing visual quality.

Each compression algorithm has its own specific features and applications. In this article we will look at different data compression algorithms, their advantages in different spheres, and discuss the key principles on which they are based.

Basic principles of data compression algorithms:

1. Redundancy removal

The first principle of data compression is to remove redundancy. Redundancy covers repetitive data and linear dependencies. Repetitive data appear when the same information occurs several times. Methods such as Lempel-Ziv-Welch method or RLE algorithm (Run-Length Encoding).

2. Replacing data with shorter representations

The second principle of data compression is to replace data with shorter representations, which involves the use of dictionaries and the replacement of characters and character sequences.

The use of dictionaries is an effective compression method, especially when there are many repetitive elements. A dictionary contains a set of unique characters or longer sequences of data and their abbreviated codes. During compression, the data is replaced with these codes from the dictionary.

The replacement of characters and character sequences is based on their frequency. Frequently occurring characters or sequences are replaced with shorter codes and rare characters or sequences are replaced with longer codes. Optimal replacements are determined using the Huffman algorithm or adaptive arithmetic coding.

3. Arithmetic coding

Arithmetic coding is a data compression method in which the entire input data stream is represented by a single number within a certain interval. In this process, each character or sequence of characters is assigned an interval of real numbers proportional to their probability. During decoding, the input stream is divided into intervals corresponding to the characters or sequences of characters, which allows the reconstruction of the original data.

4. Lossy compression

The last principle of data compression involves the use of lossy compression techniques. In such cases, some data quality may be lost, but a higher compression ratio is achieved. Examples of lossy compression algorithms are JPEG for images and MP3 for audio.

One of the key points in data compression is choosing the right algorithm for the task. Let's consider two main categories of data compression algorithms: lossless and lossy algorithms.

Lossless data compression algorithms are designed to reduce the size of data without changing its content. They rely on detecting and exploiting patterns and repetitive structures in the data.

1. Huffman algorithm

The algorithm is based on creating an optimal prefix code for each character in the original data. Prefix code is an encoding system, where each symbol corresponds to a unique code word, which is not a prefix for another code word. Such feature of coding allows to compress data efficiently without loss of information at their subsequent decoding.

2. Lempel-Ziv-Welch algorithm (LZW)

The LZW algorithm is based on the idea of forming a dictionary containing strings that have already occurred in the input data. The process starts by initialising the dictionary with basic characters, such as single characters or short strings. LZW

then looks at the input data and compares the current string with the already present dictionary entries.

If the current string is already present in the dictionary, LZW moves to the next character and repeats the process. If the current string is not present in the dictionary, LZW adds it to the dictionary and replaces the current string with a reference to a previous matching dictionary entry. This reduces the amount of data by replacing repeated sequences with shorter references.

The LZW algorithm continues scanning and building the dictionary until it reaches the end of the input data or the dictionary size limit. The resulting references and final dictionary are then used to reconstruct the original data by performing the reverse process, decompression.

Lossy data compression algorithms are used in cases where a small loss of information may be acceptable to achieve greater data reduction. These techniques are used extensively in areas where space saving is critical, such as in multimedia and audio-video compression.

1. JPEG

The JPEG algorithm is based on the concept of dividing an image into frequencies and quantising them. For this purpose, the Discrete Cosine Transform (DCT) is used, which translates the spatial representation of an image into its frequency representation. Each block of image pixels is represented by the amplitudes of different frequency components, from low to high. After applying the DCT, each block is quantised, which removes the less important frequencies with greater accuracy. The algorithm is based on the use of quantisation tables, where each table defines a quantum value for the corresponding frequency components. These tables have different quantisation levels, which reduces the amount of data while preserving details.

After quantisation, the data undergoes further Huffman coding to further reduce the file size. This method is based on using shorter codes for more frequent characters and longer codes for rare characters, which ensures efficient data compression.

2. MP3

The MP3 algorithm for lossy audio compression uses a psychoacoustic model to analyse the properties of audio signals and their human perception. Then, based on this information, the algorithm removes the less noticeable components of the audio file that are not subjectively perceived by humans. This makes it possible to significantly reduce the size of the audio file without compromising the sound quality when playing it.

In conclusion, data compression algorithms are an integral part of modern information technology, providing more efficient use of resources and increasing data transfer rates. They represent an important tool for optimising data storage and transmission, and their application can be highly beneficial for various domains, including Internet services, multimedia applications and information systems.

ДОДАТОК Е

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008: 2015

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-1 (група)

Жихарський Петро Олександрович
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	

зауважень немає

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

14.06.2024