Seminar Kompressionsalgorithmen

LZMA (Lempel-Ziv-Markov-Algorithmus): 7z

Arvid Butting

Theoretical Computer Science

25.April 2012

Motivation

- Verlustfreie Kompression beliebiger Daten

Motivation

- Verlustfreie Kompression beliebiger Daten
- Gute Kompressionsraten

Motivation

- Verlustfreie Kompression beliebiger Daten
- Gute Kompressionsraten
- Schnelles Entpacken

Motivation

- Verlustfreie Kompression beliebiger Daten
- Gute Kompressionsraten
- Schnelles Entpacken
- Geringer Speicherverbrauch bei Entpacken

Grundlagen: Bereichskodierer

- Implementierung des arithmetischen Kodierens mit Integerwerten

Grundlagen: Bereichskodierer

- Implementierung des arithmetischen Kodierens mit Integerwerten
- Patentfrei

Grundlagen: Bereichskodierer

- Implementierung des arithmetischen Kodierens mit Integerwerten
- Patentfrei
- Entropiekodierung

Grundlagen: Bereichskodierer

- Implementierung des arithmetischen Kodierens mit Integerwerten
- Patentfrei
- Entropiekodierung
- Entfernen alphabetischer Redundanz und kontextueller Redundanz

Grundlagen: Bereichskodierer

- Implementierung des arithmetischen Kodierens mit Integerwerten
- Patentfrei
- Entropiekodierung
- Entfernen alphabetischer Redundanz und kontextueller Redundanz

Funktionsweise

Grundlagen: Bereichskodierer

- Implementierung des arithmetischen Kodierens mit Integerwerten
- Patentfrei
- Entropiekodierung
- Entfernen alphabetischer Redundanz und kontextueller Redundanz

Funktionsweise

- Eingabe: $w \in \Sigma^*$ Eingabewort, $N \in \mathbb{N}$ Obergrenze und *prob* Wahrscheinlichkeit

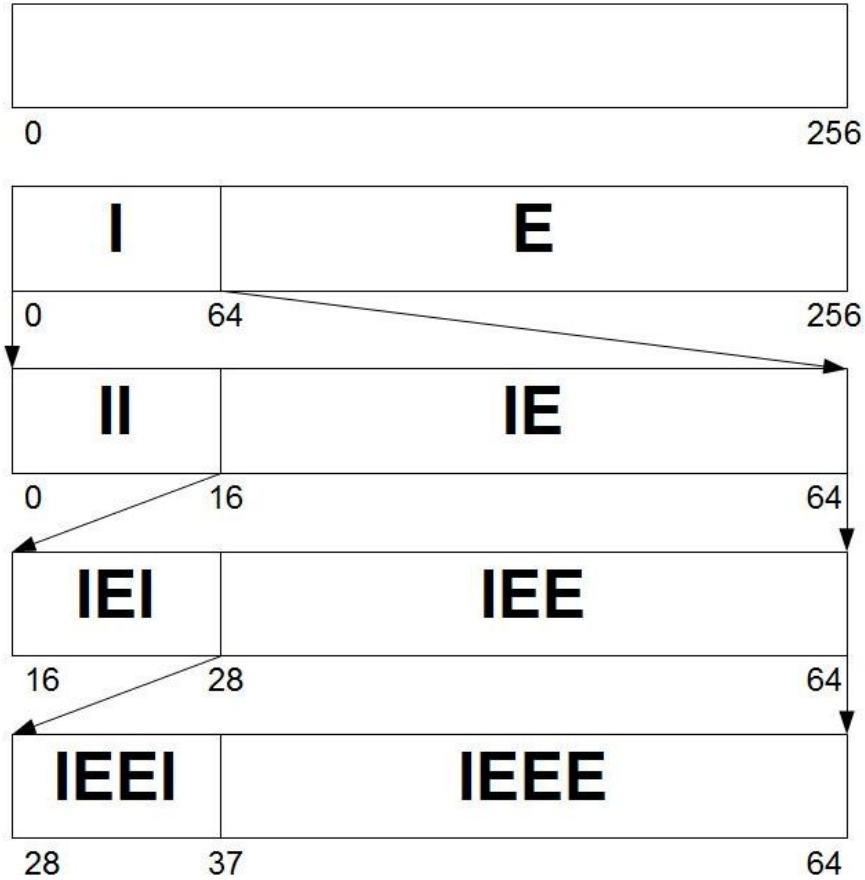
Grundlagen: Bereichskodierer

- Implementierung des arithmetischen Kodierens mit Integerwerten
- Patentfrei
- Entropiekodierung
- Entfernen alphabetischer Redundanz und kontextueller Redundanz

Funktionsweise

- Eingabe: $w \in \Sigma^*$ Eingabewort, $N \in \mathbb{N}$ Obergrenze und *prob* Wahrscheinlichkeit
- Ausgabe: $[low, range)$, wobei $low, range \in \mathbb{N}_0, low \leq range \leq N$. $c \in [low, range)$ beliebige Zahl im Intervall, die w eindeutig festlegt

Beispiel Dezimal



Binäre Version

Variablen:

- low = 0 00000000 00000000 00000000 00000000; (33 Bit)

Binäre Version

Variablen:

- low = 0 00000000 00000000 00000000 00000000; (33 Bit)
- range = 11111111 11111111 11111111 11111111; (32 Bit)

Binäre Version

Variablen:

- low = 0 00000000 00000000 00000000 00000000; (33 Bit)
- range = 11111111 11111111 11111111 11111111; (32 Bit)
- cache = 00000000; (8 Bit)

Binäre Version

Variablen:

- low = 0 00000000 00000000 00000000 00000000; (33 Bit)
- range = 11111111 11111111 11111111 11111111; (32 Bit)
- cache = 00000000; (8 Bit)
- cacheSize = 0..0 0..0 0..0 0..0 0..0 0..0 0..0 0..01; (64 Bit)

Binäre Version

Variablen:

- low = 0 00000000 00000000 00000000 00000000; (33 Bit)
- range = 11111111 11111111 11111111 11111111; (32 Bit)
- cache = 00000000; (8 Bit)
- cacheSize = 0..0 0..0 0..0 0..0 0..0 0..0 0..0 0..01; (64 Bit)
- prob = 100 00000000 (11 Bit)

Funktionsweise

Enkodieren

if($range < 2^{24}$)
    normalisiere();

$bound = (range \gg 11)$ * $prob$)

if($bitToCode == 0$)
    $range = bound$;
    $prob = prob + (2^{11}$ - $prob) \gg 5$;

if($bitToCode == 1$)
    $range = range$ - $bound$;
    $low = low + bound$;
    $prob = prob$ - $(prob \gg 5)$;

Terminiere nach 5 Normalisierungen

Normalisieren

if($low < (2^{32} - 2^{24})$)
    output($cache$);
    output1Bytes($cacheSize - 1$);
    $cache$ = sel24To31OfLow();
    $cacheSize = 0$;

if($low \geq (2^{32})$)
    output($cache$+1);
    output0Bytes($cacheSize - 1$);
    $cache$ = sel24To31OfLow();
    $cacheSize = 0$;

$cacheSize++$;
$low = (low \ \& \ (2^{24} - 1)) \ll 8$;
$range = range \ll 8$;

Funktionsweise

Enkodieren

if($range < 2^{24}$)
    normalisiere();

$bound = (range \gg 11)$ * $prob$)

if($bitToCode == 0$)
    $range = bound$;
    $prob = prob + (2^{11}$ - $prob) \gg 5$;

if($bitToCode == 1$)
    $range = range$ - $bound$;
    $low = low + bound$;
    $prob = prob$ - $(prob \gg 5)$;

Terminiere nach 5 Normalisierungen

Normalisieren

if($low < (2^{32} - 2^{24})$)
    output($cache$);
    output1Bytes($cacheSize - 1$);
    $cache$ = sel24To31OfLow();
    $cacheSize = 0$;

if($low \geq (2^{32})$)
    output($cache$+1);
    output0Bytes($cacheSize - 1$);
    $cache$ = sel24To31OfLow();
    $cacheSize = 0$;

$cacheSize++$;
$low = (low \ \& \ (2^{24} - 1)) \ll 8$;
$range = range \ll 8$;

▷ Beispiel auf Projektor

Funktionsweise

Dekodieren

if($range < 2^{24}$)
    normalisiere();

$bound = (range \gg 11)$ * $prob$)

if($code < bound$)
    $range = bound$;
    $prob = prob + (2^{11}$ - $prob) \gg 5$;
    return 0;

else
    $range = range$ - $bound$;
    $code = code$ - $bound$;
    $prob = prob$ - $(prob \gg 5)$;
    return 1;

Normalisieren

$range = range \ll 8$;
$code = code \ll 8$;
$code = code +$ readByte();

Funktionsweise

Dekodieren

if($range < 2^{24}$)
    normalisiere();

$bound = (range \gg 11)$ * $prob$)

if($code < bound$)
    $range = bound$;
    $prob = prob + (2^{11}$ - $prob) \gg 5$;
    return 0;

else
    $range = range$ - $bound$;
    $code = code$ - $bound$;
    $prob = prob$ - $(prob \gg 5)$;
    return 1;

Normalisieren

$range = range \ll 8$;
$code = code \ll 8$;
$code = code +$ readByte();

▷ Beispiel auf Projektor

Geschichte von LZMA

1998: Erste veröffentlichte Version von Igor Pavlov

Geschichte von LZMA

1998: Erste veröffentlichte Version von Igor Pavlov

2000: Erste Verwendung von LZMA im Dateiformat 7z

Geschichte von LZMA

1998: Erste veröffentlichte Version von Igor Pavlov

2000: Erste Verwendung von LZMA im Dateiformat 7z

2008: LZMA SDK veröffentlicht

Geschichte von LZMA

1998: Erste veröffentlichte Version von Igor Pavlov

2000: Erste Verwendung von LZMA im Dateiformat 7z

2008: LZMA SDK veröffentlicht

2008: LZMA2 veröffentlicht

Funktionsweise

Ungefähre Beschreibung

"LZMA ist ein Variante von LZ77 mit großem Wörterbuch, deren Ausgabe mit einem Bereichskodierer kodiert wird."

Funktionsweise

Ungefähre Beschreibung

"LZMA ist ein Variante von LZ77 mit großem Wörterbuch, deren Ausgabe mit einem Bereichskodierer kodiert wird."

1. Wörterbuchkompression, Größe bis zu 4GB

Funktionsweise

Ungefähre Beschreibung

"LZMA ist ein Variante von LZ77 mit großem Wörterbuch, deren Ausgabe mit einem Bereichskodierer kodiert wird."

1. Wörterbuchkompression, Größe bis zu 4GB
2. Outputstreams in Pakete eingeteilt

Funktionsweise

Ungefähre Beschreibung

"LZMA ist ein Variante von LZ77 mit großem Wörterbuch, deren Ausgabe mit einem Bereichskodierer kodiert wird."

1. Wörterbuchkompression, Größe bis zu 4GB
2. Outputstreams in Pakete eingeteilt
   - Literale

Funktionsweise

Ungefähre Beschreibung

"LZMA ist ein Variante von LZ77 mit großem Wörterbuch, deren Ausgabe mit einem Bereichskodierer kodiert wird."

1. Wörterbuchkompression, Größe bis zu 4GB
2. Outputstreams in Pakete eingeteilt
   - Literale
   - LZ77 Sequenzen

Funktionsweise

Ungefähre Beschreibung

"LZMA ist ein Variante von LZ77 mit großem Wörterbuch, deren Ausgabe mit einem Bereichskodierer kodiert wird."

1. Wörterbuchkompression, Größe bis zu 4GB
2. Outputstreams in Pakete eingeteilt
   - Literale
   - LZ77 Sequenzen
   - Schnellzugriff auf zuletzt vorkommende LZ77 Sequenzen

Funktionsweise

Ungefähre Beschreibung

"LZMA ist ein Variante von LZ77 mit großem Wörterbuch, deren Ausgabe mit einem Bereichskodierer kodiert wird."

1. Wörterbuchkompression, Größe bis zu 4GB
2. Outputstreams in Pakete eingeteilt
   - Literale
   - LZ77 Sequenzen
   - Schnellzugriff auf zuletzt vorkommende LZ77 Sequenzen
3. Berechnen von Wahrscheinlichkeiten für Bereichskodierer

Funktionsweise

Ungefähre Beschreibung

"LZMA ist ein Variante von LZ77 mit großem Wörterbuch, deren Ausgabe mit einem Bereichskodierer kodiert wird."

1. Wörterbuchkompression, Größe bis zu 4GB
2. Outputstreams in Pakete eingeteilt
   - Literale
   - LZ77 Sequenzen
   - Schnellzugriff auf zuletzt vorkommende LZ77 Sequenzen
3. Berechnen von Wahrscheinlichkeiten für Bereichskodierer
4. Ausgabe bitweise mit Bereichkodierer kodiert

Wörterbuch

Vorgehensweise mit Hashkette

Wörterbuch

Vorgehensweise mit Hashkette

1. Hashen von 2-4 Bytes (im folgenden w) im Vorschaupuffer

Wörterbuch

Vorgehensweise mit Hashkette

1. Hashen von 2-4 Bytes (im folgenden w) im Vorschaupuffer
2. Ausgabe der Hashfunktion ist Index i eines Hasharrays ($hash[i]$) von Listen

Wörterbuch

Vorgehensweise mit Hashkette

1. Hashen von 2-4 Bytes (im folgenden *w*) im Vorschaupuffer
2. Ausgabe der Hashfunktion ist Index *i* eines Hasharrays (*hash[i]*) von Listen
   - An *hash[i]* stehen alle bisherigen Matches von *w*

Wörterbuch

Vorgehensweise mit Hashkette

1. Hashen von 2-4 Bytes (im folgenden *w*) im Vorschaupuffer
2. Ausgabe der Hashfunktion ist Index *i* eines Hasharrays (*hash[i]*) von Listen
   - An *hash[i]* stehen alle bisherigen Matches von *w*
3. Die ersten 24 Einträge werden überprüft, der optimalste wird ausgewählt

Wörterbuch

Vorgehensweise mit Hashkette

1. Hashen von 2-4 Bytes (im folgenden *w*) im Vorschaupuffer
2. Ausgabe der Hashfunktion ist Index *i* eines Hasharrays (*hash[i]*) von Listen
   - An *hash[i]* stehen alle bisherigen Matches von *w*
3. Die ersten 24 Einträge werden überprüft, der optimalste wird ausgewählt
4. Anschließend weitergereicht und kodiert

Wörterbuch

Mögliche Datenstrukturen:

- Hashkette

Wörterbuch

Mögliche Datenstrukturen:

- Hashkette
- PATRICIA-Trie (nicht mehr verwendet)

Wörterbuch

Mögliche Datenstrukturen:

- Hashkette
- PATRICIA-Trie (nicht mehr verwendet)
- Binärer Suchbaum

Wörterbuch

Mögliche Datenstrukturen:

- Hashkette
- PATRICIA-Trie (nicht mehr verwendet)
- Binärer Suchbaum

Kriterien des binären Suchbaums

Wörterbuch

Mögliche Datenstrukturen:

- Hashkette
- PATRICIA-Trie (nicht mehr verwendet)
- Binärer Suchbaum

Kriterien des binären Suchbaums

- Gültiger binärer Suchbaum, lexikographisch geordnet

Wörterbuch

Mögliche Datenstrukturen:

- Hashkette
- PATRICIA-Trie (nicht mehr verwendet)
- Binärer Suchbaum

Kriterien des binären Suchbaums

- Gültiger binärer Suchbaum, lexikographisch geordnet
- Gültiger Max-Heap bezüglich der Wörterbuchposition

Wörterbuch

Mögliche Datenstrukturen:

- Hashkette
- PATRICIA-Trie (nicht mehr verwendet)
- Binärer Suchbaum

Kriterien des binären Suchbaums

- Gültiger binärer Suchbaum, lexikographisch geordnet
- Gültiger Max-Heap bezüglich der Wörterbuchposition

▷ Beispiel auf Projektor

Pakettypen

Kodierung	**Name**	**Bedeutung**
0 + byteCode	LIT	1 Byte Literal
10 + length+dist	MATCH	LZ77 Sequenz
1100	SREP	Exakte Wiederholung des letzten Matches
1101 + length	REP[0]	Distanz wie letzter Match
1110 + length	REP[1]	Distanz wie vorletzter Match
11110 + length	REP[2]	Distanz wie drittletzter Match
11111 + length	REP[3]	Distanz wie viertletzter Match

Transitionssystem

Übergangsfunktion

Zustand	LIT	MATCH	REP[*]	SREP
0	0	7	8	9
1	0	7	8	9
2	0	7	8	9
3	0	7	8	9
4	1	7	8	9
5	2	7	8	9
6	3	7	8	9
7	4	10	11	11
8	5	10	11	11
9	6	10	11	11
10	4	10	11	11
11	5	10	11	11

- Beispiel auf Projektor

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt
- Containerformat für LZMA-komprimierte und unkomprimierte Blöcke

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt
- Containerformat für LZMA-komprimierte und unkomprimierte Blöcke
  1. Jeder Block wird LZMA-komprimiert

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt
- Containerformat für LZMA-komprimierte und unkomprimierte Blöcke
  1. Jeder Block wird LZMA-komprimiert
  2. Größen des unkomprimiertem und komprimiertem Blocks werden verglichen

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt
- Containerformat für LZMA-komprimierte und unkomprimierte Blöcke
  1. Jeder Block wird LZMA-komprimiert
  2. Größen des unkomprimiertem und komprimiertem Blocks werden verglichen
  3. Kleinerer wird in die Datei geschrieben

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt
- Containerformat für LZMA-komprimierte und unkomprimierte Blöcke
  1. Jeder Block wird LZMA-komprimiert
  2. Größen des unkomprimiertem und komprimiertem Blocks werden verglichen
  3. Kleinerer wird in die Datei geschrieben
- LZMA Blöcke evtl. in verschiedenen Konfigurationen kodiert

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt
- Containerformat für LZMA-komprimierte und unkomprimierte Blöcke
  1. Jeder Block wird LZMA-komprimiert
  2. Größen des unkomprimiertem und komprimiertem Blocks werden verglichen
  3. Kleinerer wird in die Datei geschrieben
- LZMA Blöcke evtl. in verschiedenen Konfigurationen kodiert
- Unterstützt Multithreading bei (De-)Kompression

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt
- Containerformat für LZMA-komprimierte und unkomprimierte Blöcke
  1. Jeder Block wird LZMA-komprimiert
  2. Größen des unkomprimiertem und komprimiertem Blocks werden verglichen
  3. Kleinerer wird in die Datei geschrieben
- LZMA Blöcke evtl. in verschiedenen Konfigurationen kodiert
- Unterstützt Multithreading bei (De-)Kompression
  - Geteiltes Wörterbuch

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt
- Containerformat für LZMA-komprimierte und unkomprimierte Blöcke
  1. Jeder Block wird LZMA-komprimiert
  2. Größen des unkomprimiertem und komprimiertem Blocks werden verglichen
  3. Kleinerer wird in die Datei geschrieben
- LZMA Blöcke evtl. in verschiedenen Konfigurationen kodiert
- Unterstützt Multithreading bei (De-)Kompression
  - Geteiltes Wörterbuch
  - Blöcke können unabhängig voneinander behandelt werden

LZMA2

- Datenstream in Blöcke (*Chunks*) unterteilt
- Containerformat für LZMA-komprimierte und unkomprimierte Blöcke
  1. Jeder Block wird LZMA-komprimiert
  2. Größen des unkomprimiertem und komprimiertem Blocks werden verglichen
  3. Kleinerer wird in die Datei geschrieben
- LZMA Blöcke evtl. in verschiedenen Konfigurationen kodiert
- Unterstützt Multithreading bei (De-)Kompression
  - Geteiltes Wörterbuch
  - Blöcke können unabhängig voneinander behandelt werden

LZMA2 liefert meist eine höhere Kompressionsrate als LZMA, besonders bei (teilweise) unkomprimierbaren Daten.

Dateiformat 7z

Dateiformate, die LZMA implementieren

Dateiformat 7z

Dateiformate, die LZMA implementieren

- 7z- Format

Dateiformat 7z

Dateiformate, die LZMA implementieren

- 7z- Format
- xz- Format (nur LZMA2)

Dateiformat 7z

Dateiformate, die LZMA implementieren

- 7z- Format
- xz- Format (nur LZMA2)

7z Format

- Entwickelt von Igor Pavlov

Dateiformat 7z

Dateiformate, die LZMA implementieren

- 7z- Format
- xz- Format (nur LZMA2)

7z Format

- Entwickelt von Igor Pavlov
- Benutzt in 7-Zip Software, aber auch unterstützt von anderer Kompressionssoftware

Dateiformat 7z

Dateiformate, die LZMA implementieren

- 7z- Format
- xz- Format (nur LZMA2)

7z Format

- Entwickelt von Igor Pavlov
- Benutzt in 7-Zip Software, aber auch unterstützt von anderer Kompressionssoftware
- Standardkompressionsmethode ist LZMA

Dateiformat 7z

Dateiformate, die LZMA implementieren

- 7z- Format
- xz- Format (nur LZMA2)

7z Format

- Entwickelt von Igor Pavlov
- Benutzt in 7-Zip Software, aber auch unterstützt von anderer Kompressionssoftware
- Standardkompressionsmethode ist LZMA
- Archivierungsfunktion

Dateiformat 7z

Dateiformate, die LZMA implementieren

- 7z- Format
- xz- Format (nur LZMA2)

7z Format

- Entwickelt von Igor Pavlov
- Benutzt in 7-Zip Software, aber auch unterstützt von anderer Kompressionssoftware
- Standardkompressionsmethode ist LZMA
- Archivierungsfunktion
- Unterstützt zudem PPMD, BZip2 und Deflate, sowie BCJ und BCJ2

Dateiformat 7z

Dateiformate, die LZMA implementieren

- 7z- Format
- xz- Format (nur LZMA2)

7z Format

- Entwickelt von Igor Pavlov
- Benutzt in 7-Zip Software, aber auch unterstützt von anderer Kompressionssoftware
- Standardkompressionsmethode ist LZMA
- Archivierungsfunktion
- Unterstützt zudem PPMD, BZip2 und Deflate, sowie BCJ und BCJ2
- bietet Verschlüsselung der Daten mit AES-256

Mögliche Parameter

Parameter	-a[N]
Bedeutung	Kompressionsmodus
Wert	$N \in 0, 1, 2$ ($\widehat{=}$ schnell, normal, max)
Default	$N = 2$

Mögliche Parameter

Parameter	-a[N]
Bedeutung	Kompressionsmodus
Wert	$N \in 0, 1, 2$ ($\widehat{=}$ schnell, normal, max)
Default	$N = 2$

Parameter	-d[N]
Bedeutung	Wörterbuchgröße
Wert	$N \in \{0, 1, \ldots, 30\}$
Default	$N = 27$

Mögliche Parameter

Parameter	-a[N]
Bedeutung	Kompressionsmodus
Wert	$N \in 0, 1, 2$ ($\widehat{=}$ schnell, normal, max)
Default	N = 2

Parameter	-d[N]
Bedeutung	Wörterbuchgröße
Wert	$N \in \{0, 1, \ldots, 30\}$
Default	N = 27

Parameter	-mf[ID]
Bedeutung	Match Finder
Wert	$ID \in \{bt2, bt3, bt4, hc4\}$
Default	ID = bt4 , nur im schnellen Modus hc4

xz Format

- Entwickelt von Lasse Collin

xz Format

- Entwickelt von Lasse Collin
- Portierung von LZMA auf Linux

xz Format

- Entwickelt von Lasse Collin
- Portierung von LZMA auf Linux
- Unterstützt nur LZMA2

xz Format

- Entwickelt von Lasse Collin
- Portierung von LZMA auf Linux
- Unterstützt nur LZMA2
- Keine Archivierungsfunktion

xz Format

- Entwickelt von Lasse Collin
- Portierung von LZMA auf Linux
- Unterstützt nur LZMA2
- Keine Archivierungsfunktion
- (De-)Komprimieren mit XZ-Utils

xz Format

- Entwickelt von Lasse Collin
- Portierung von LZMA auf Linux
- Unterstützt nur LZMA2
- Keine Archivierungsfunktion
- (De-)Komprimieren mit XZ-Utils
- XZ Embedded in Linux Kernel integriert

7-Zip

- Entwickelt von Igor Pavlov

7-Zip

- Entwickelt von Igor Pavlov
- Grafische Oberfläche und Konsole

7-Zip

- Entwickelt von Igor Pavlov
- Grafische Oberfläche und Konsole
- nur für Windows erhältlich

7-Zip

- Entwickelt von Igor Pavlov
- Grafische Oberfläche und Konsole
- nur für Windows erhältlich
- unterstützt neben LZMA und LZMA2 viele weitere Kompressionsmethoden

7-Zip

- Entwickelt von Igor Pavlov
- Grafische Oberfläche und Konsole
- nur für Windows erhältlich
- unterstützt neben LZMA und LZMA2 viele weitere Kompressionsmethoden
- Erreicht oftmals bessere Kompression als andere Kompressionsprogramme

7-Zip

- Entwickelt von Igor Pavlov
- Grafische Oberfläche und Konsole
- nur für Windows erhältlich
- unterstützt neben LZMA und LZMA2 viele weitere Kompressionsmethoden
- Erreicht oftmals bessere Kompression als andere Kompressionsprogramme
- Open Source unter der GNU-LGPL Lizenz (mit wenigen Ausnahmen)

Vergleich von Kompressionsraten

Text: (*Faust, J.W. von Goethe*[1])
Bild: (*7Zip Logo*[2])
zufälliger Bitstream(*20MB, mit Math.random() in Java*)

	Umkompr.	.7z (LZMA)	.zip (Deflate)	.bzip2 (BWT)
Text:	222 kB	75.9 kB	82.1 kB	69 kB
Bild:	192 kB	7.52 kB	11.6 kB	10 kB
zufällig:	20480 kB	19077 kB	18867 kB	19809 kB

[1] Quelle: http://www.gutenberg.org/files/21000/21000-0.txt
[2] Quelle: http://www.articlevoid.com/wp-content/uploads/7zip.png