Kompresní techniky

David Bařina

13. února 2019

Obsah

Co to jsou multimédia?

Co to jsou multimédia?



zabývají se zvukem, obrázky, animacemi, videem, ... a dalšími daty



Úvod

- multimediální data: text, zvuk, obraz, animace, video, ...
- jeden snímek FullHD videa $\approx 6\,\mathrm{MB}$ (při $60\,\mathrm{FPS} \approx 360\,\mathrm{MB/s}$)
- jedna sekunda 4K videa $\approx 1{,}5\,\mathrm{GiB}$
- 2 hod. 4K videa $\approx 10{,}5\,\mathrm{TiB}$
- formáty: BMP, PNG, JPEG, MPEG-4, ...

Základní pojmy

Komprese

- zmenšení objemu dat, datového toku
- bezeztrátová vs. ztrátová
- snížení redundance vs. irelevance dat

Redundance (nadbytečnost)

- zmenšení redundance = bezeztrátová komprese
- původní signál lze zrekonstruovat bez zkreslení

Irelevance

- z hlediska vnímání člověkem
- odstranění irelevantních dat = ztrátová komprese
- nelze přesně zrekonstruovat původní signál

Základní pojmy

Kódování

- vzájemně jednoznačné přiřazení symbolů jedné abecedy symbolům abecedy druhé
- např. symbol 'A' $\rightarrow$ kód 0110
- snížení redundance = komprese

Počítačový program nebo hardware

- kompresor/kodér
- dekompresor/dekodér
- kodek (audiokodek, videokodek)
- formát dat, např. formát audia, formát videa

DIVX™ $\neq$ MPEG 4

Základní pojmy

Model dat

- pravděpodobnostní (statistický) model
- kontextový (Markovský) model

Kompresní metody

- symetrická vs. asymetrická (složitost)
- bloková vs. proudová (blok symbolů), např. bzip2 $\sim$900 kB
- jednoprůchodová, dvouprůchodová, víceprůchodová

- statická = 1 průchod, model dat stanoven
- semiadaptivní = 2 průchody, model třeba přenést
- adaptivní (dynamická) = 1 průchod, model upravován za běhu

Základní pojmy

Kódování s proměnnou délkou (VLC)

- krátké kódy pro často se vyskytující symboly
- např. 'A' $\rightarrow$ 1, 'B' $\rightarrow$ 01, . . . , 'Z' $\rightarrow$ 000000001
- realizuje tzv. entropické kódování (dále)
- např. Huffmanovy kódy

Prefixový kód

- vlastnost, že žádný kód (kódové slovo) není prefixem jiného kódu
- při dekódování jednoznačný bez oddělovačů
- např. 1, 01, 001, 0001, . . .

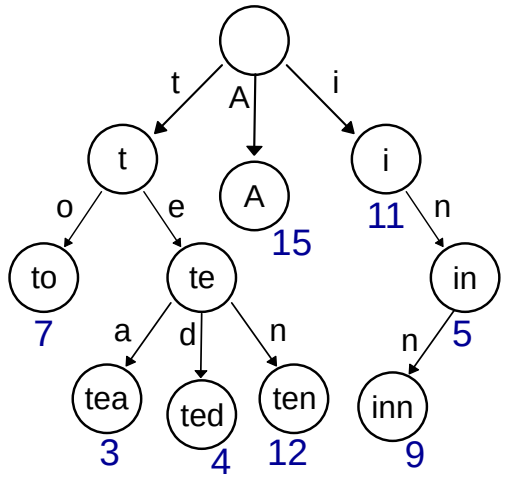
Základní pojmy

Slovník

- datová struktura obsahující fragmenty nekomprimovaného souboru
- např. 0 → "aa", 1 → "ba", 2 → "bc"
  10021 → "baaaaabcba"

Kompresní metody

- statistické (statistický model dat)
- kontextové
- slovníkové (slovník)



Základní pojmy

Kompresní poměr

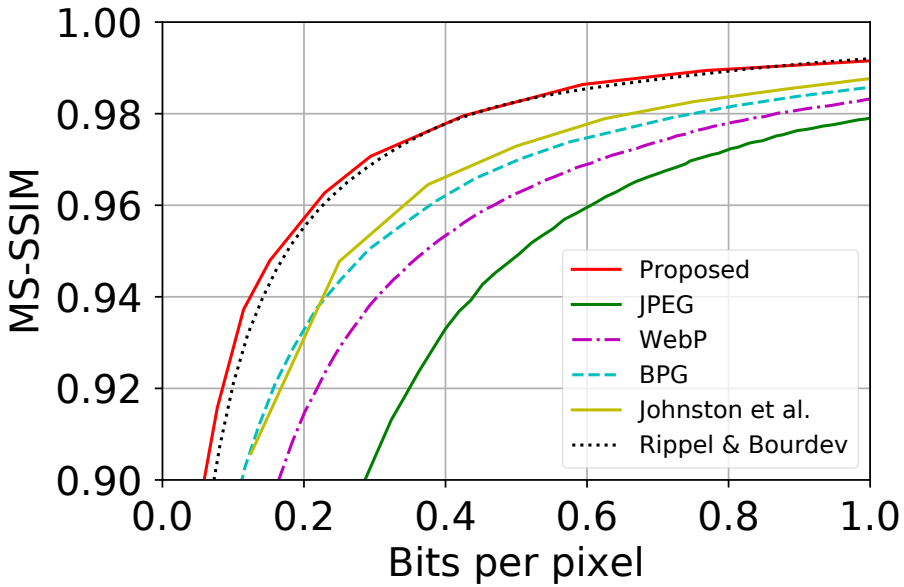
$$= \frac{\text{velikost výstupu}}{\text{velikost vstupu}}$$

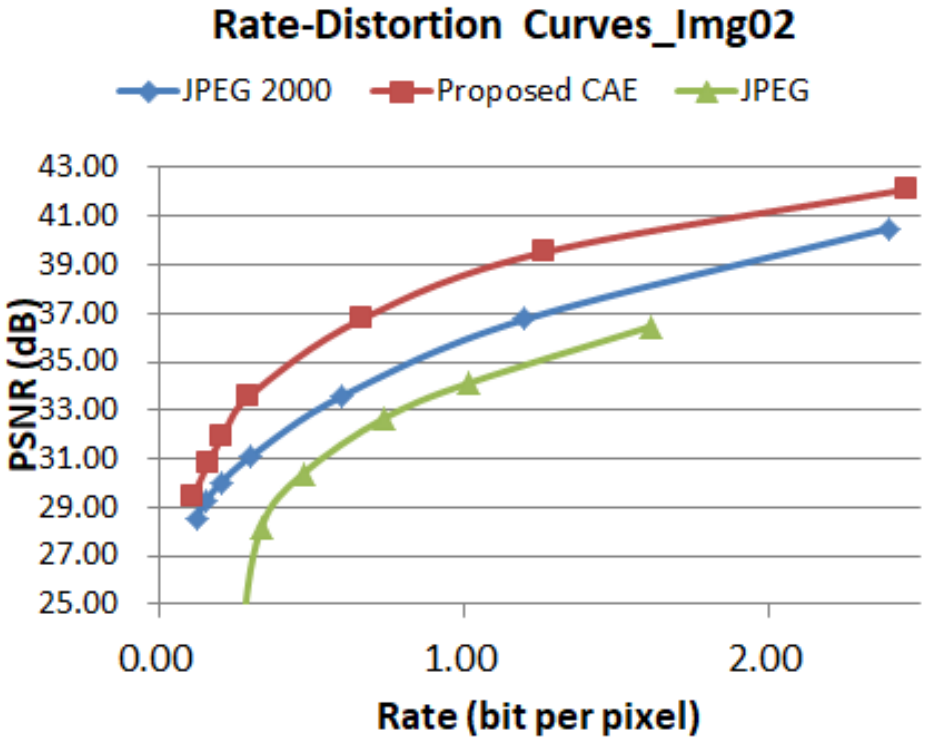
($< 1 \Rightarrow$ komprese, $> 1 \Rightarrow$ expanze)

Hodnocení kompresní metody

- účinnost (kompresní poměr)
- časová, paměťová náročnost
- vliv typu dat na kompresní poměr
- u ztrátových metod kvalita, často PSNR
- závislost kvality na kompresním poměru (rate–distortion curve)

Rate–distortion (RD) curve





Základní pojmy

Kompresní metody

- prediktivní (prediktor, chyba)
- transformační (např. DCT)

Entropie dat

- veličina udávající množství „informace“
- míra překvapení, informační obsah, potřebný počet bitů
- počet otázek ano/ne, kterými lze odhalit obsah zprávy
- jednotkou bit

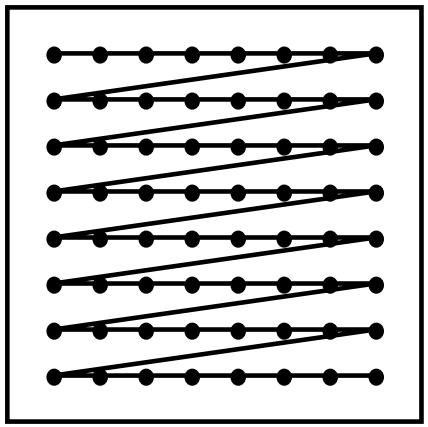
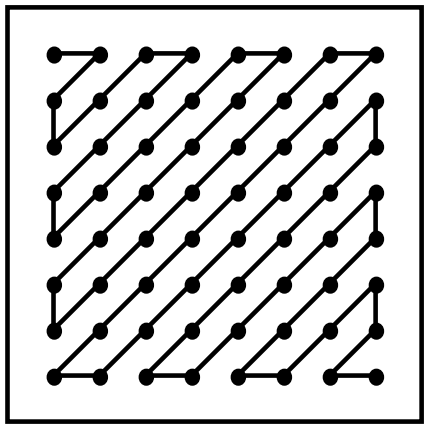
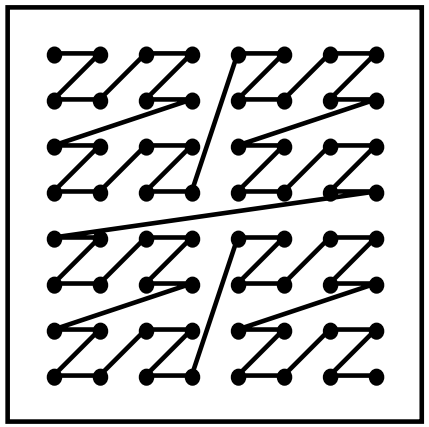
$$H = -\sum_{a \in A} p(a) \log_2 p(a)$$

- entropické kodéry komprimují data
  téměř optimálně vzhledem k jejich entropii

Vícerozměrná data

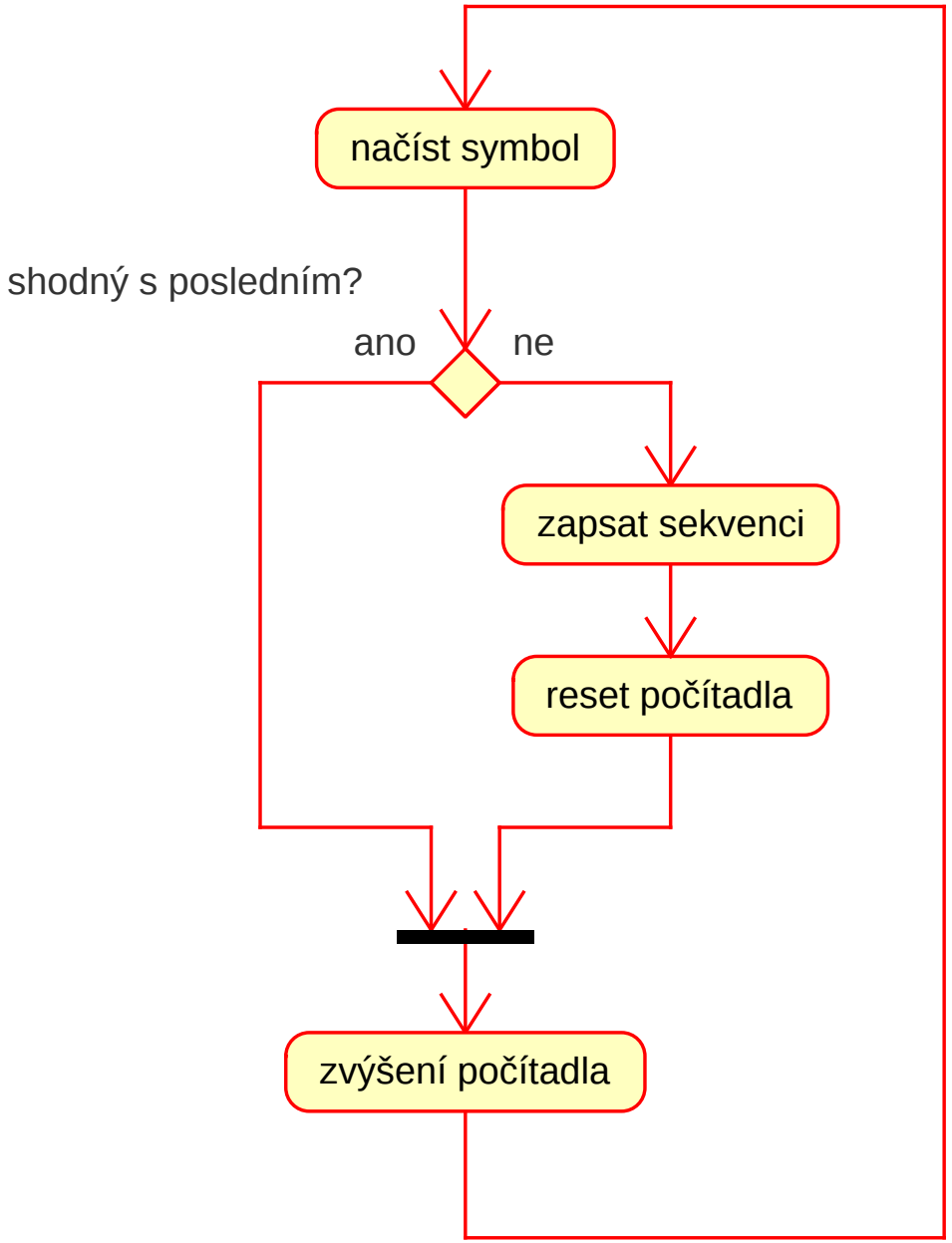
Linearizační průchody

- rastrový průchod (po řádcích)
- zig-zag, např. DCT u JPEG
- Mortonův průchod (Z-křivka), např. strom DWT

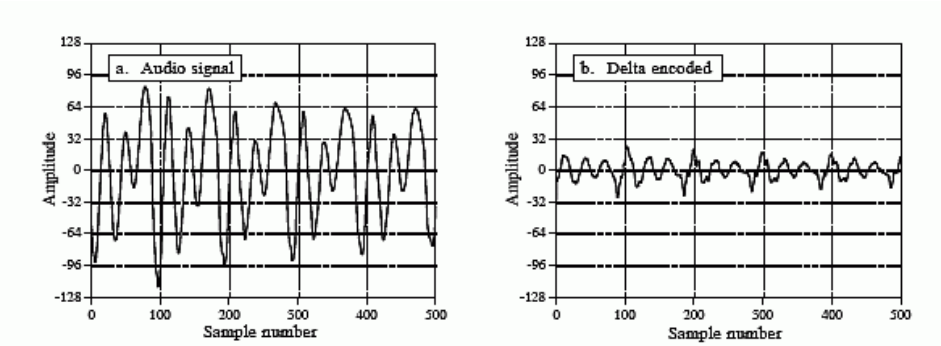
RLE (Run-Length Encoding)

- kódování sledů (posloupností, řady) stejných znaků
- mnoho modifikací
- součást složitějších metod
- kód nese informaci o zakódovaném symbolu a počtu jeho opakování
- např. „`A A A A B C D`“ → „`4×A B C D`“
- „escape“ symbol, „escape“ sekvence
- použit téměř všude (BMP, PCX, TIFF, TGA, JPEG, bzip2)



Kódování rozdílů

- relativní kódování, delta kódování
- součást složitějších metod
- nahrazení vstupních hodnot za rozdíly od hodnot předcházejících
- jednoduchá predikční metoda
- dále se kóduje chyba predikce
- např.
  17 16 18 32 35 35 34 28 28 $\rightarrow$ −1 +2 +14 +3 0 −1 −6 0



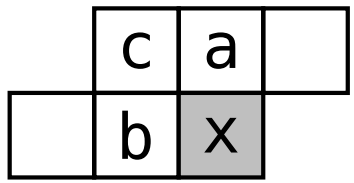
Prediktivní kódování

- kódovaný symbol se predikuje z předchozích symbolů
- dále se kóduje chyba predikce
- řády prediktorů (1., 2., 3., . . .)
- domény (1D, 2D, strom)

Prediktory

- lineární (kódování rozdílů, Lossless JPEG, PNG)
- nelineární (medián, MED/LOCO-I, Paeth, GAP)

$$\hat{x} = \begin{cases} \min(a,b) & : c \geq \max(a,b) \\ \max(a,b) & : c \leq \min(a,b) \\ a+b-c & \end{cases}$$



Unární kódování

- velmi jednoduché entropické kodování
- optimální pro $p(n) = 2^{-n}$
- mapuje nezáporná celá čísla N na bitové kódy, např. $N \rightarrow N \times 1, 0$

  $0 \rightarrow 0$
  $1 \rightarrow 10$
  $2 \rightarrow 110$
  $3 \rightarrow 1110$
  $\ldots$
- 0 a 1 lze zaměnit

Golombovo-Riceovo kódování

- speciální případ Golombova kódování; rychlá implementace
- užity např. v JPEG-LS, FLAC, MPEG-4 ALS
- mapuje nezáporná celá čísla N na bitové kódy
- kód je parametrizovatelný parametrem $M = 2^C$
- pro $M = 1$ se jedná o unární kódování
- pro získání kódu čísla N se nejprve určí

$$Q = \lfloor N/M \rfloor \quad R = N - Q \cdot M \quad C = \lceil \log_2 M \rceil$$

- Q se pak kóduje unárně, R binárně na C bitech

Golombovo-Riceovo kódování

pro $M = 4$

N	Q	R	kód
0	0	0	1 00
1	0	1	1 01
2	0	2	1 10
3	0	3	1 11
4	1	0	01 00
5	1	1	01 01
6	1	2	01 10
7	1	3	01 11
8	2	0	001 00
9	2	1	001 01
10	2	2	001 10
11	2	3	001 11
12	3	0	0001 00

Shannonovo–Fanovo kódování

- metoda vytváří kódová slova podle pravděpodobnosti výskytu kódovaných symbolů (adaptuje se na data)
- nejlepších výsledků dosahuje, když jsou pravděpodobnosti záporné mocniny 2
- Huffmanovo kódování v praxi generuje o něco lepší kód
- symboly jsou listy v binárním stromě, jehož hrany (0 a 1) reprezentují kód symbolu
- konstrukce stromu:
  1. seřadit symboly sestupně dle jejich pravděpodobností
  2. rozdělit tuto množinu na dvě tak, že obě budou mít nejlépe stejný součet pravděpodobností
  3. rekurzivně aplikovat krok 2. na obě podmnožiny (uzly stromu) až do rozkladu na jednotlivé symboly
- použito v ZIP/Implode

Shannonovo–Fanovo kódování

příklad

pravděpodobnost					kód
0,25	1	1			11
0,20	1	0			10
0,15	0	1	1		011
0,15	0	1	0		010
0,10	0	0	1		001
0,10	0	0	0	1	0001
0,05	0	0	0	0	0000

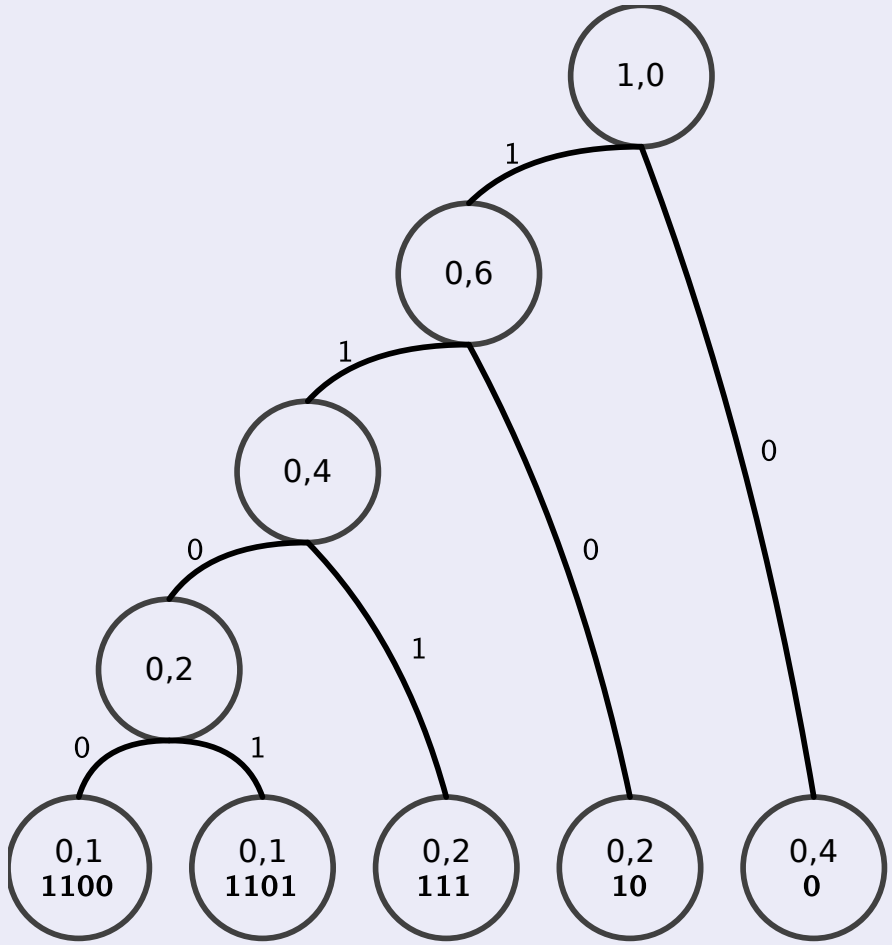
Huffmanovo kódování

- populární metoda
- adaptuje se na data
- nejlepší výsledky pro pravděpodobnosti rovny záporné mocnině 2
- symboly jsou listy v binárním stromě s hranami udavajícími kód
- použití: bzip2, Deflate, JPEG, MP3
- konstrukce stromu:
  1. seřadit symboly sestupně dle jejich pravděpodobností
  2. vyber 2 symboly s nejnižší pravděpodobností, spoj je do nového uzlu
  3. pokračuj krokem 2. dokud je co spojovat
- u adaptivní varianty je nutné opravovat strom

Huffmanovo kódování

příklad

p					kód
0,4				0	0
0,2			0	1	10
0,2		1	1	1	111
0,1	1	0	1	1	1101
0,1	0	0	1	1	1100



Aritmetické kódování

- optimální kódy pro libovolné pravděpodobnosti výskytu symbolů
- metoda přidělí jeden kód celému kódovanému souboru dat
- začíná se s intervalem, který se podle pravděpodobností kódovaných symbolů neustále zužuje
- zužování intervalu vyžaduje další bity, takže délka kódu postupně roste
- myšlenka komprese: symbol s vyšší pravděpodobností zúží interval méně (přidá méně bitů) než symbol s pravděpodobností nižší
- praktické implementace musejí pracovat s celými čísly
- použití: kontextové kodéry, JPEG, Dirac

Aritmetické kódování

příklad

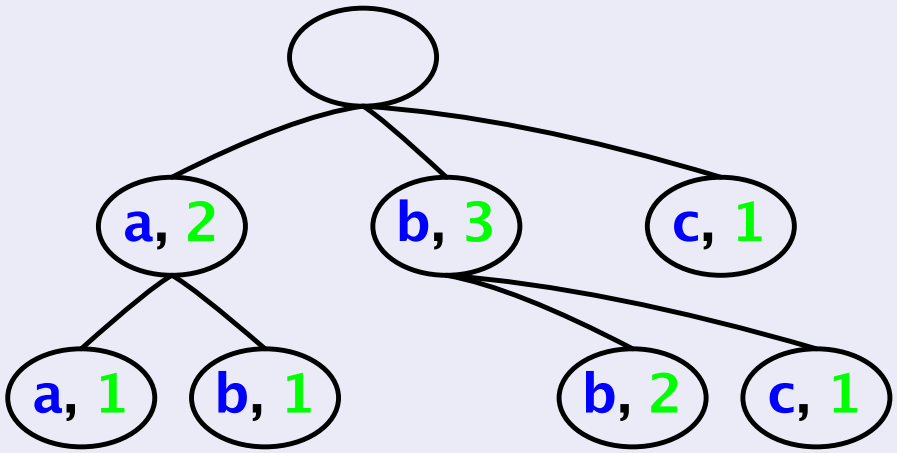
Kontextová komprese

- využívají nějčastěji arit. kódování nebo jeho modifikaci
- na rozdíl od něj nekódují pravděpodobnost výskytu osamoceného symbolu, ale pravděpodobnost jeho výskytu v určitém kontextu
- kontext: několik předcházejích symbolů, okolních pixelů, okolních bitů, sousedních koeficientů $\rightarrow$ řád kontextu
- kontext musejí tvořit již přenesené symboly
- kontext využit k predikci (přidělení pravděpodobnosti výskytu symbolu)
- escape kódy: pokud nelze z kontextu pravděpodobnost určit (PPMx)
- použití: MPEG-4 (CABAC, CAVLC), JPEG 2000 (EBCOT), JPEG-LS, PPMx

Kontextová komprese

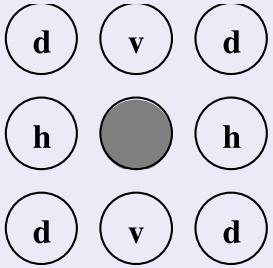
příklad

- řetězec aabbbc
- kontext řádu 1
- tento model predikuje např. po symbolu 'b' symbol 'b' s pravděpodobností 66 % a symbol 'c' s pravděpodobností 33 %



EBCOT

kodér použitý
ve standardu JPEG 2000



LZ77

- publikovali A. Lempel a J. Ziv v roce 1977
- mnoho modifikací
- použita v Deflate
- pohyblivé okno (sliding window)
- dvě části: vyhledávácí a předvídací buffer
- v praxi tisíce vs. desítky bajtů
- tvoří značky `(offset, délka, symbol)`

←...`east#easily#t`|`eases`...←

- 2× shoda „eas“ na pozicích 8 a 13
- vytvoří se značka `(13, 3, 'e')`
- prvky značky se zakódují na odpovídajícím počtu bitů $\lceil \log_2 S \rceil, \lceil \log_2 (L-1) \rceil, \lceil \log_2 A \rceil$, kde A je velikost abecedy

LZ77

←...|east#easily#t|rashe|...←

- při nenalezení shody se generuje značka `(0, 0, 'r')`

←...|east#easily#t|ttttt|...←

- shoda může překročit hranici vyhledávácího bufferu, zde `(1, 5, 't')`
- to je také důvod pro počet bitů $\lceil \log_2(L-1) \rceil$ délky
- LZ77 předpokládá, že fragmenty se vyskytují blízko u sebe
- existuje množství vylepšení této metody: značky proměnné délky, důmyslné datové struktury, zrušení posledního pole značky, např. metoda LZSS

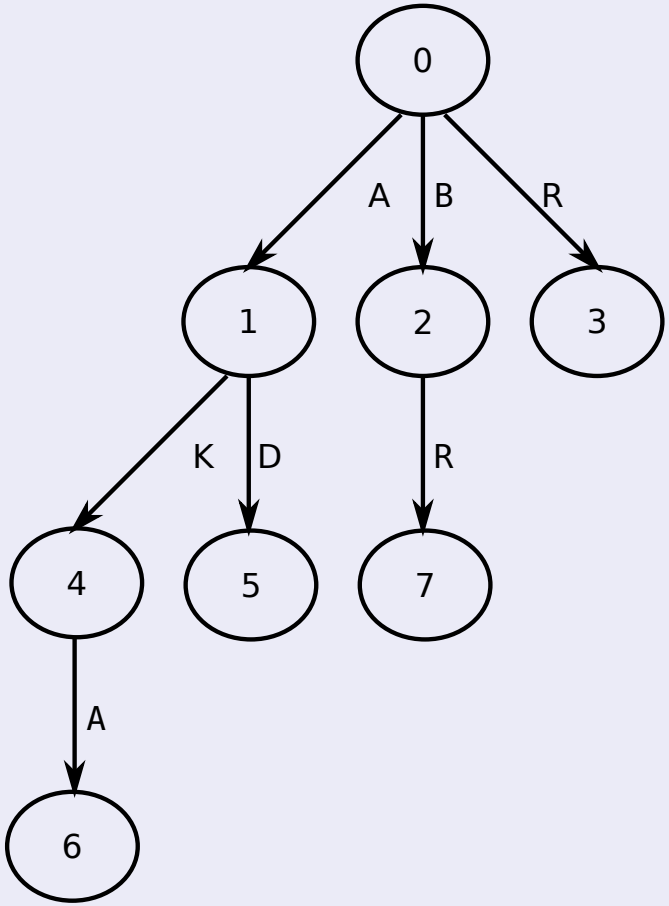
LZ78

- publikovali A. Lempel a J. Ziv v roce 1978
- mnoho modifikací, např. LZW
- nemá pohyblivé okno, ale slovník
- je náročná na paměť, lze řešit různě
- vhodná datová struktura k udržování slovníku je trie
- první položka slovníku je prázdný řetězec
- kodér vytváří značky `(index, symbol)`
- při kompresi se vyhledá nejdelší shodný řetězec ve slovníku a vygeneruje se značka s jeho indexem a následujícím symbolem
- každá značka udává řetězec, ten se umístí do slovníku

LZ78

Příklad: ABRAKADAKABRA

- na počátku prázdný slovník
- 'A', 'B', 'R' se zakódují jako
  `(0, 'A'), (0, 'B'), (0, 'R')`
- 'AK' a 'AD' se přidají pod uzel 'A' a zakódují se jako
  `(1, 'K'), (1, 'D')`
- stejně se zakóduje AKA, BR a poslední A jako
  `(4, 'A'), (2, 'R'), (1, EOF)`



LZW

- varianta LZ78, kterou vyvinul T. Welch v roce 1984
- použit u GIF, TIFF, PDF
- použit slovník, na počátku inicializován na všechny symboly
- značka má pouze jedno pole: `(index)`
- kodér:
  1. ve vstupu hledá nejdelší řetězec obsažený ve slovníku I
  2. následující znak x způsobí, že Ix ve slovníku již není
  3. na výstup nyní vyšle index I, uloží Ix do slovníku, I nyní bude jen x
  4. pokračuje se krokem 1.
- dekodér:
  1. načte index a zapíše odpovídající řetězec I na výstup
  2. je třeba přidat do slovníku řetězec Ix, ale x ještě není známo
  3. načte se další index, na výstup zapíše odpovídající řetězec J, jeho první znak je x
  4. nyní může uložit Ix do slovníku, J bude dále I
  5. pokračuje se bodem 2.

LZW: příklad

Příklad: `sir#sid#` (kodér)

vstup x=’s’	Ix=’s’ nalez.	I=Ix=’s’		
vstup x=’i’	Ix=’si’ nenalez.	výstup idx. I=’s’	ulož Ix=’si’	I=x=’i’
vstup x=’r’	Ix=’ir’ nenalez.	výstup idx. I=’i’	ulož Ix=’ir’	I=x=’r’
vstup x=’#’	Ix=’r#’ nenalez.	výstup idx. I=’r’	ulož Ix=’r#’	I=x=’#’
vstup x=’s’	Ix=’#s’ nenalez.	výstup idx. I=’#’	ulož Ix=’#s’	I=x=’s’
vstup x=’i’	Ix=’si’ nalez.	I=Ix=’si’		
vstup x=’d’	Ix=’sid’ nenalez.	výstup idx. I=’si’	ulož Ix=’sid’	I=x=’d’
vstup x=’#’	Ix=’d#’ nenalez.	výstup idx. I=’d’	ulož Ix=’d#’	I=x=’#’
vstup x=EOF	Ix=’#EOF’ nenalez.	výstup idx. I=’#’	konec	

Slovník:

0–255	znaky 0–255
256	’si’
257	’ir’
258	’r#’
259	’#s’
260	’sid’
261	’d#’

LZW: příklad

Příklad: `sir#sid#` (dekodér)

vstup index J='s'	výstup J='s'	x=J(1)='s'	ulož Ix='s'	I=J='s'
vstup index J='i'	výstup J='i'	x=J(1)='i'	ulož Ix='si'	I=J='i'
vstup index J='r'	výstup J='r'	x=J(1)='r'	ulož Ix='ir'	I=J='r'
vstup index J='#'	výstup J='#'	x=J(1)='#'	ulož Ix='r#'	I=J='#'
vstup index J='si'	výstup J='si'	x=J(1)='s'	ulož Ix='#s'	I=J='si'
vstup index J='d'	výstup J='d'	x=J(1)='d'	ulož Ix='sid'	I=J='d'
vstup index J='#'	výstup J='#'	x=J(1)='#'	ulož Ix='d#'	I=J='#'

Slovník:

0–255	znaky 0–255
256	'si'
257	'ir'
258	'r#'
259	'#s'
260	'sid'
261	'd#'

Deflate

- použita v ZIP (původně), zlib/gzip, 7-Zip, PNG, MNG, PDF
- kombinace LZ77 a Huffmanova kódování
- na rozdíl od LZ77 mají značky jen dvě pole `(offset, délka)`
- scházející položka `(symbol)` se zapisuje do výstupního toku zvlášť
- komprimovaný tok se tedy skládá ze tří entit: literálů/symbolů, offsetů/vzdáleností a délek
- tyto entity se kódují pomocí Huffmanových kódů za pomoci dvou tabulek: literály+délky a vzdálenosti
- délky jsou omezeny na 258, literály jsou bajty (0–255); vzdálenost až do velikosti bufferu 32 kB
- data komprimuje po blocích (odděleně) různé délky

Deflate

- definuje 3 režimy komprese:
  1. bez komprese (max. 65 535 B)
  2. komprese s fixními tabulkami
  3. komprese s tabulkami zapsanými do komprimovaného toku
- výběr shody je odložen

←...`she#needs#then#there#`|`the#new`...←

- nevybere (1,3) = „the“
- „t“ zakóduje jako literál
- vybere (20,5) = „he#ne“

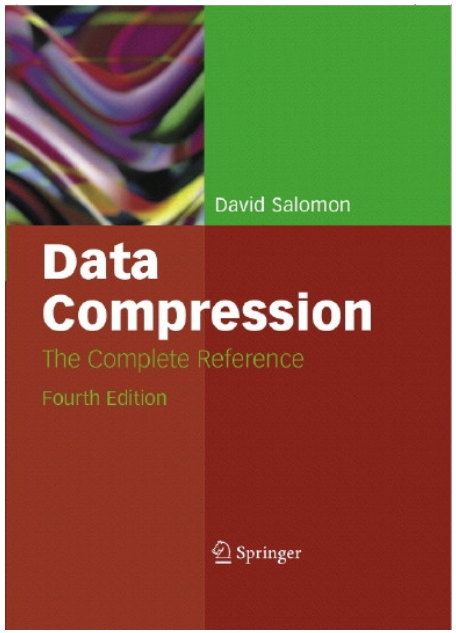
Používané kombinace metod

- data (nekomprimovaná)
- data $\rightarrow$ RLE (BMP, TGA)
- data $\rightarrow$ predikce $\rightarrow$ EC (JPEG-LS)
- data $\rightarrow$ predikce $\rightarrow$ slovníková metoda (PNG)
- data $\rightarrow$ slovníková metoda (GIF)
- data $\rightarrow$ transformace $\rightarrow$ RLE+EC (JPEG)
- data $\rightarrow$ transformace $\rightarrow$ kontextový EC (JPEG 2000, MPEG)

Opakování

- pojmy (redundance, model dat, dělení kompresních metod, prefixový kód, slovník, kompresní poměr, entropie)
- zpracování vícerozměrných dat
- základní metody: RLE, kódování rozdílů, predikce
- entropické kodéry (unární, Golombovo-Riceovo, Huffmanovo a aritmetické kódování)
- kontextová komprese
- slovníkové metody (LZ77, LZ78, LZW a Deflate)
- kombinace metod

Zdroje



- David Salomon. Data Compression: The Complete Reference. 4. ilustrované vydání, Springer, 2006.
- specifikace formátů PNG, JPEG, JPEG-LS, JPEG 2000, ...