Bezstratová kompresia údajov
aritmetické kódovanie

ÚINF/KMÚ ZS 2019/20
doc. RNDr. Jozef Jirásek, PhD.

Aritmetické kódovanie - princíp

Aritmetické kódovanie kóduje celú vstupnú sekvenciu znakov x_1 x_2 ... x_k n-znakovej abecedy zdroja jediným kódovým slovom (číslom z intervalu [0,1)).

Začíname s celým intervalom [0,1).
Pre každý znak x_i vstupnej sekvencie postupne rozdelíme interval podľa pravdepodobností $p_1, p_2, \ldots, p_n$ výskytov znakov abecedy v zdrojovom texte a ďalej pracujeme už len s podintervalom, príslušným vstupu x_i .

Kódom bude číslo z intervalu, ktorý sme takto po k krokoch dostali (napr. číslo s najkratším binárnym zápisom).

Počet bitov kódu závisí od veľkosti intervalu, preto sekvencia s vyššou pravdepodobnosťou dostane kratší kód.

Aritmetické kódovanie - príklad

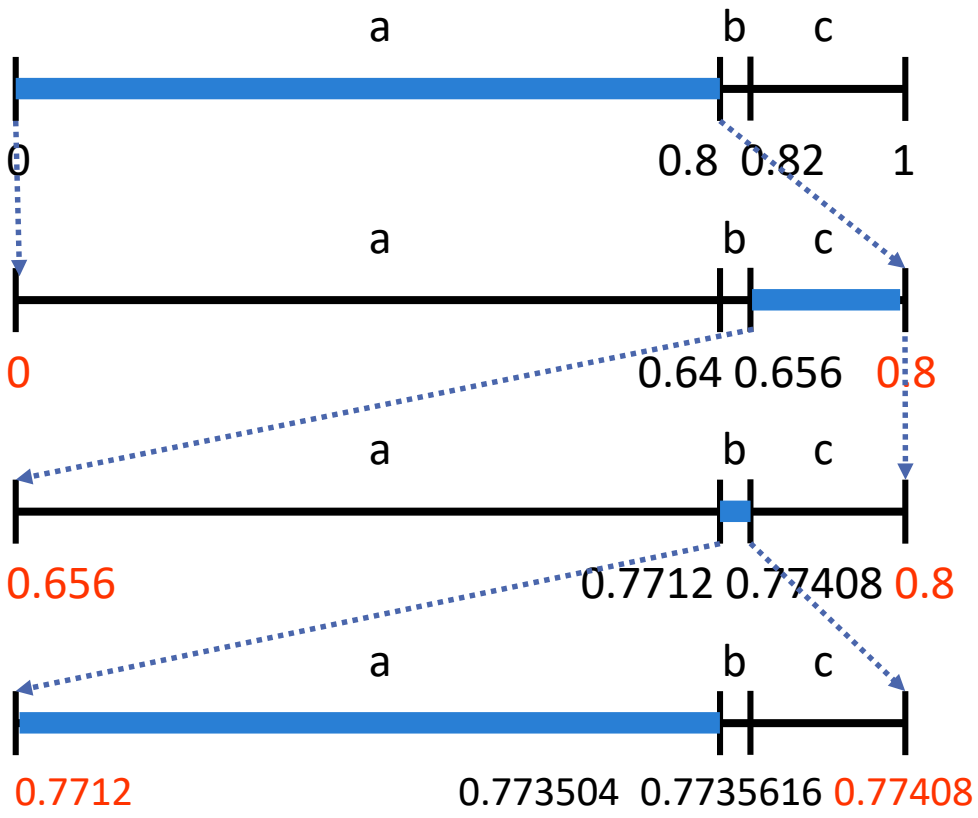
vstup „acba“

Symbol	Pravdep.
a	0.8
b	0.02
c	0.18

disjunktné intervaly

- a: [0, 0.8)
- b: [0.8, 0,82)
- c: [0.82, 1)

poradie musí poznať aj dekóder
(napr. podľa abecedy)
musí poznať aj dĺžku vstupu !



Kódový interval pre vstup acba : [0.7712, 0.773504) kódovanie : 0.7712

Aritmetické kódovanie – algoritmus kódovania

- distribučná funkcia $F_X(i) = \sum_{j=1}^{i} p_j$ pre i-ty znak abecedy;
  $F_X(0) = 0$ (CDF – cumulative distribution function)
- $l_0 = 0; h_0 = 1; r_0 = 1;$
- ak vstupný znak x_i je m-ty znak abecedy, potom
  $l_i = l_{i-1} + r_{i-1}F_X(\mathrm{m}-1);\ \ h_i = l_{i-1} + r_{i-1}F_X(m);$
  $r_i = h_i - l_i = r_{i-1}(F_X(m) - F_X(\mathrm{m}-1)) = r_{i-1}p_m;$
- pre vstup $x_1, x_2, \ldots, x_k$ dostaneme interval $[l_k, h_k)$
  a kódom (*tag*) bude $T(x_1x_2\ldots x_k) = (l_k + h_k)/2$ (resp. l_k)
- $r_k = p(x_k) \cdot p(x_{k-1}) \cdot \ldots \cdot p(x_1) = p(x_1x_2\ldots x_k)$
  rozsah posledného intervalu $[l_k, h_k)$ zodpovedá
  pravdepodobnosti výskytu celého vstupného textu

Aritmetické kódovanie – dekódovanie

tag T = 0.7712

je v intervale [0, 0.8)
- dekódujeme a

prepočítame hranice

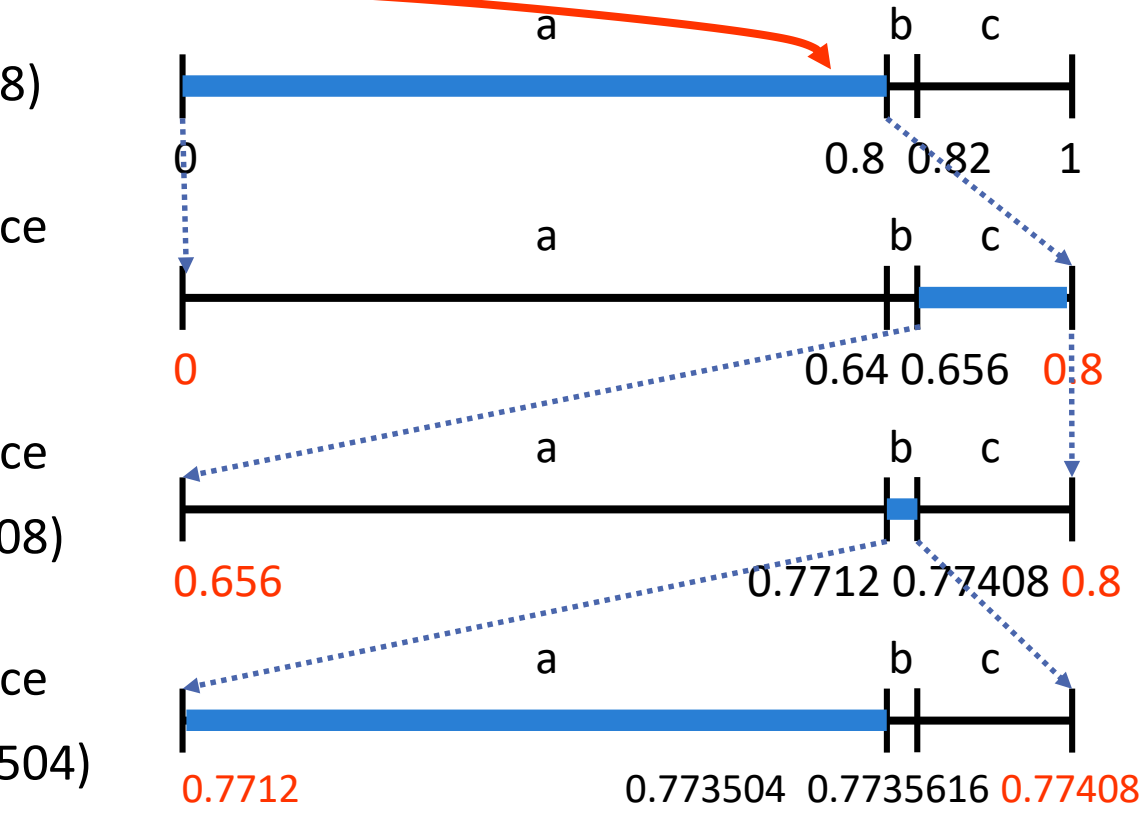
T ∈ [0.656, 0.8)
- dekódujeme c

prepočítame hranice

T ∈ [0.7712, 0.77408)
- dekódujeme b

prepočítame hranice

T ∈ [0.7712, 0.773504)
- dekódujeme a



Aritmetické kódovanie – zjednodušenie

jednoduchšie je prepočítať tag do nového rozsahu

tag $T_0 = 0.7712$

je v intervale [0, 0.8)
- dekódujeme a

$T_1 = (T_0 - 0)/0.8 = 0.964$

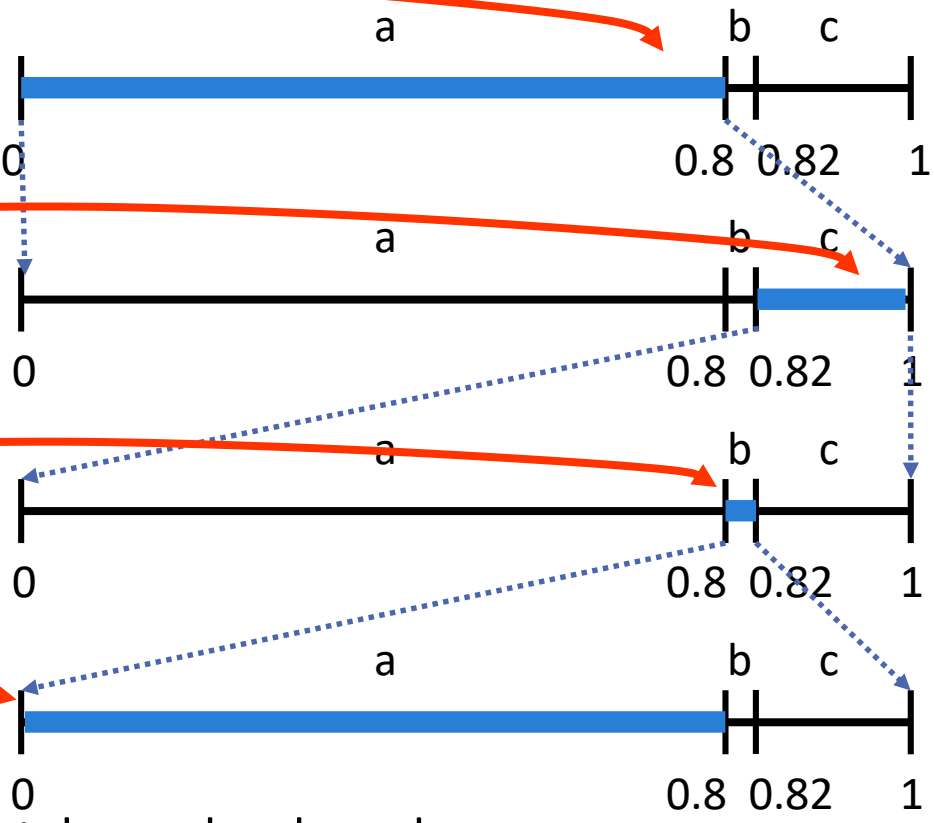
je v intervale [0.82, 1)
- dekódujeme c

$T_2 = (T_1 - 0.82)/0.18 = 0.8$

0.8 je v intervale [0.8, 0.82)
- dekódujeme b

$T_3 = (T_2 - 0.8)/0.02 = 0$

- dekódujeme a



všeobecne po dekódovaní m-teho znaku abecedy

$$T_i = (T_{i-1} - F_X(\mathrm{m} - 1))/p_m$$

Aritmetické kódovanie – zjednodušenie

zjednodušenie kódovania – pokiaľ mám k dispozícii celý vstupný text

zo vzťahu $T_i = (T_{i-1} - F_X(\mathrm{m}-1))/p_m$

platí $T_{i-1} = T_i p_m + F_X\,(\mathrm{m}-1)$ a teda ak kódujeme priemerom $T_k = 1/2$

pre i od k po 1 spočítame $T_{i-1} = T_i p_{ord(xi)} + F_X\,(ord(x_i) - 1)$

kde $ord(x_i)$ je poradové číslo znaku x_i vstupu

T_0 bude hľadaný tag

Kódové slovo pre binárny výstup

- najkratšia sekvencia bitov $b_1 b_2 \dots b_m$ taká, že binárne číslo $0.b_1 b_2 \dots b_m$ je v kódovom intervale
- pre $r = p(x_1 x_2 \dots x_k)$ musí byť veľkosť intervalu $r \geq \frac{1}{2^m}$ (aby bol rozlíšiteľný od ostatných) $\Rightarrow \quad m \geq \lceil -\log r \rceil$
- na kódové slovo budeme teda potrebovať aspoň $\lceil -\log r \rceil$ bitov z tagu $T = T(x_1 x_2 \dots x_k) = (l_k + h_k)/2$ (pre stred intervalu najviac $\lceil -\log r \rceil + 1$)
- $L_k = \sum_{i=1}^{k} p_i l_i \leq \sum_{i=1}^{k} p_i (\lceil -\log p_i \rceil + 1) <$
  $< \sum_{i=1}^{k} p_i (-\log p_i + 2) = H(S^k) + 2$
- $Rate = L < (H(S^k) + 2)/k = H(S) + 2/k$
- závisí len od dĺžky vstupu

Inkrementálne kódovanie

- aby sme nemuseli pracovať s veľmi veľkou presnosťou výpočtu, priebežne posielame výstupný kód (ak je začiatok intervalu stabilný)
- E_1 mapovanie ak $h_i < 0.5$ (výstup 0)
  $$E_1 : [0, 0.5) \rightarrow [0, 1): \; E_1(x) = 2x$$
- E_2 mapovanie ak $l_i \geq 0.5$ (výstup 1)
  $$E_2 : [0.5, 1) \rightarrow [0, 1): \; E_2(x) = 2(x - 0.5)$$

Inkrementálne kódovanie

vstup	$l_i = l_{i-1} + r_{i-1}F_X(\mathrm{m}-1)$	$h_i = l_{i-1} + r_{i-1}F_X(m)$	$r_i = r_{i-1}p_m;$	výstup
init	0	1	1	
a (0,8)	0+1*0 = 0	0+1*0,8 = 0,8	1*0,8 = 0,8	
c (0,18)	0+0,8*0,82 = 0,656	0+0,8*1 = 0,8	0,8*0,18 = 0,144	1
rescale	0,312	0,6	0,288	
b (0,02)	0,312+0,288*0,8 = 0,5424	0,312+0,288*0.82 = 0,54816	0,288*0,02 = 0,00576	1
rescale	0,0848	0,09632	0,01152	0
rescale	0,1696	0,19264	0,02304	0
rescale	0,3392	0,38528	0,04608	0
rescale	0,6784	0,77056	0,09216	1
rescale	0,3568	0,54112	0,18432	
a (0,8)	0,3568	0,504256	0,147456	

Kódové slovo

- priebežný výstup 110001
- interval (0.3568, 0.504256) – možno vybrať 0.5 = $(0.1)_2$ teda výstup bude 0,110001 1 ~ 0,7734365
- presnejšie - možno vybrať z (0.0101101..., 0.10000001...)

- dekódovanie ... postupné dopĺňanie bitov sprava
- E_3 mapovanie ak $l_i \geq 0.25$ a $h_i < 0.75$
  $E_3: [0.25, 0.75) \rightarrow [0, 1)$: $E_3(x) = 2(x - 0.25)$

  počíta sa počet mapovaní E_3 a pri najbližšom výskyte mapovania E_1 (resp. E_2) sa na výstup pošle po bite 0 (resp. 1) napočítaný počet (za E_3) opačných bitov 1 (resp. 0)

Aritmetické celočíselné kódovanie

práca len s celými číslami

- odstráni sa problém nejednoznačného zaokrúhľovania
- rýchlejší výpočet s ohraničenou presnosťou
- inkrementálne kódovanie – výstup sa generuje priebežne

rozsah m bitov 0000...0 až 1111...1 [0, 1)
(mapovanie 0,5 = 100...0)

škálovanie (najvyššie bity v binárnej reprezentácii možno odoslať)

- problém – zmenšovanie intervalu s rozdielnymi najvyššími bitmi 01111... 1000... vyriešime škálovaním E_3

Aritmetické celočíselné kódovanie

namiesto pravdepodobností – početnosti $q_1, q_2, \ldots, q_n$

namiesto distribučnej funkcie - kumulatívny súčet pre i-ty znak abecedy $Q(i) = \sum_{j=1}^{i} q_j \quad \mathrm{Q}(0) = 0$

totálny súčet $S = Q(n) = \sum_{j=1}^{n} q_j$

- $l_0 = 0; h_0 = 111..111$
- ak vstupný znak x_i je r-ty znak abecedy, potom
  $l_i = l_{i-1} + \lfloor (h_{i-1} - l_{i-1} + 1) \cdot Q(r-1)/S \rfloor$
  $h_i = l_{i-1} + \lfloor (h_{i-1} - l_{i-1} + 1) \cdot Q(\mathrm{r})/S \rfloor - 1$
- aby sa dal realizovať každý interval početností, musí byť $S < 2^m/4$ teda $m > \lceil \log 4S \rceil = 2 + \lceil \log S \rceil$

Škálovanie binárne

MSB (most significant bit) = najvyšší bit slova
LSB (least significant bit) = najnižší bit slova

- E_1 : ak $MSB(l_i) = MSB(h_i) = 0$
  výstup 0; posun vľavo; $LSB(l_i) = 0;\ LSB(h_i) = 1$
- E_2 : ak $MSB(l_i) = MSB(h_i) = 1$
  výstup 1; posun vľavo; $LSB(l_i) = 0;\ LSB(h_i) = 1$
- E_3 : ak l_i začína 01 a h_i začína 10
  zvýšiť počítadlo; posun vľavo;
  $LSB(l_i) = MSB(l_i) = 0; LSB(h_i) = MSB(h_i) = 1$
- v prípade E_1 resp. E_2 ak je počítadlo nenulové – výstup príslušného počtu opačných bitov a vynulovanie počítadla

Aritmetické celočíselné kódovanie

	i	p_i	q_i	Q(i)
	0	-	-	0
a	1	0.8	40	40
b	2	0.02	1	41
c	3	0.18	9	50

$$S = 50$$

$$l_i = l_{i-1} + \lfloor (h_{i-1} - l_{i-1} + 1) \cdot Q(r-1)/S \rfloor$$

$$h_i = l_{i-1} + \lfloor (h_{i-1} - l_{i-1} + 1) \cdot Q(\mathrm{r})/S \rfloor - 1$$

vstup	l_i	h_i	výstup
init	00000000	11111111	
a	0+⌊256 ∗ 0/50⌋ = 0	0+⌊256 ∗ 40/50⌋-1 = 203	
c	0+⌊204 ∗ 41/50⌋ = 167 (10100111)	0+⌊204 ∗ 50/50⌋-1 = 203 (11001011)	1
rescale	01001110 = 78	10010111 = 151	E_3/1
rescale	00011100 = 28	10101111 = 175	
b	28+⌊148 ∗ 40/50⌋ = 146 (10010010)	28+⌊148 ∗ 41/50⌋-1 = 148 (10010100)	10/0
rescale	00100100	00101001	0010
rescale	01000000	10011111	E_3/1
rescale	00000000 = 0	10111111 = 191	
a	0+⌊192 ∗ 0/50⌋ = 0	0+⌊192 ∗ 40/50⌋-1 = 152	010000000

Aritmetické celočíselné dekódovanie

- kódové slovo 1100010010000000
- t = 11000100 $l_0 = 0;\ h_0 = 11111111;$ Q = (0, 40, 41, 50); S = 50
- výstup podľa hranice Q, ktorú dosiahne hodnota $\left\lfloor \frac{(t-l_i+1)S-1}{h_i-l_i+1} \right\rfloor$
- $l_i\ h_i$ ako pri kódovaní, pri škálovaní posunieme aj t a sprava doplníme pokračovanie kódového slova
  (v prípade E_3 ešte negujeme všetky nové MSB – pre $l\ h$ aj t)

t	l_i	h_i	$\left\lfloor \frac{(t-l_i+1)S-1}{h_i-l_i+1} \right\rfloor$	výstup
11000100	00000000	11111111	38	a
11000100	0	203	48	c
11000100	167 (10100111)	203 (11001011)		
10001001	01001110 = 78	10010111 = 151		
10010010/1	00011100 = 28	10101111 = 175	40	b
10010010	146 (10010010)	148 (10010100)	0	a

Adaptívne aritmetické celočíselné kódovanie

- začíname s početnosťami 1 pre každý znak
- po spracovaní vstupného znaku vždy upravíme početnosti, čo spôsobí zmenu veľkostí intervalov
- hodnota S nesmie prekročiť 2^{r-1} – pri hraničnej hodnote sa početnosti preškálujú (predelia 2) (žiadna by nemala klesnúť na nulu)

Binárne aritmetické kódovanie

- kódujeme binárne vstupy
- adaptívna verzia sleduje početnosti výskytov 0 a 1 vo vstupnom reťazci + škálovanie
- príjemca postupne pri dekódovaní jednotlivých bitov počíta početnosti tiež

Binárny Q kóder

- MPS – More Probable Symbol, LPS – Less Probable Symbol
- závisí na zdroji informácie (čierne na bielom, biele na čiernom)
- AC : $l_i = l_{i-1} + r_{i-1}F_X(\mathrm{m}-1);\ \ r_i = r_{i-1}p_m;\ p = (1-Q, Q)$

Q – pravdepodobnosť LPS, C – dolný limit (0), A – veľkosť (1)

kódovanie MPS: A := A*(1-Q);

LPS: C := C+A*(1-Q); A := A*Q

škálovanie = ak A<0.5 posun vľavo (po LPS, niekedy aj po MPS)

výstup = (vysunuté bity z C) + C

dekódovanie ak C<A*(1-Q) : A := A*(1-Q); out MPS

inak : C :=C-A*(1-Q); A := A*Q; out LPS

- M – kóder – namiesto násobenia A*Q prehľadáva tabuľku
- adaptívna verzia - Q sa mení podľa frekvencie znakov vstupu

Binárny QM kóder

- odstránenie násobenia aproximáciou
- Q – pravdepodobnosť LPS; C = 0, A = 1

kódovanie MPS: A := A - Q;

LPS: C := C + A - Q; A := Q;

ak klesne A pod 0.5, škálujeme A:=A*2; C:=C*2

výstup = (vysunuté bity z C) + C

dekódovanie ak C < A - Q : A := A - Q; out MPS

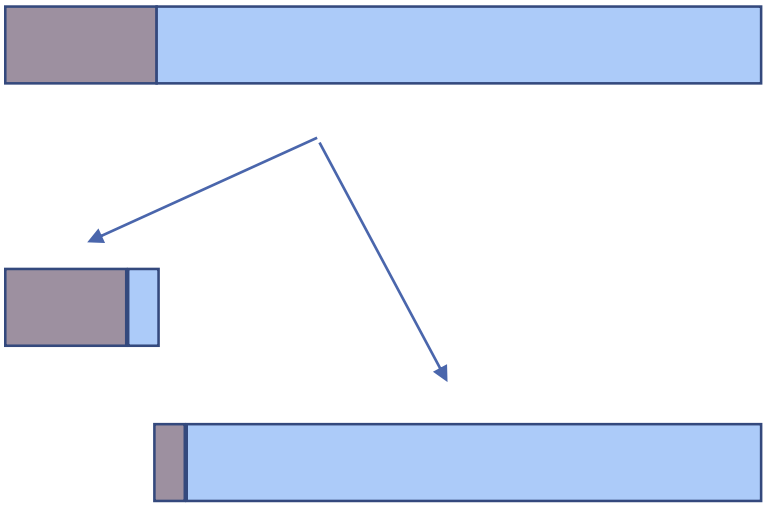
inak : C := C – A + Q; A := Q; out LPS

ak klesne A pod 0.5, škálujeme A:=A*2; C:=C*2 + bit kódu

dekódujeme až kým C=0

- adaptívna verzia - Q sa mení podľa frekvencie znakov vstupu
- ak by Q malo prekročiť 0.5 – mení sa význam MPS a LPS

Vplyv kontextu na kompresiu



p(0) = 0.2 p(1) = 0.8

H = 0.722

R ≈ 0.722 b/px AC

rozdelíme na dve skupiny

20% kde p(0) = 0.7

H = 0.881

80% kde p(1) = 0.925

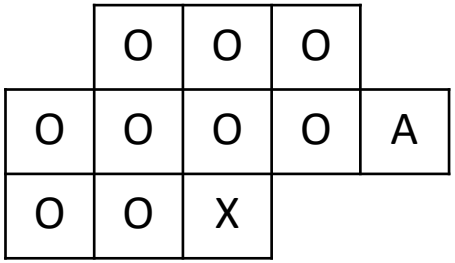
H = 0.384

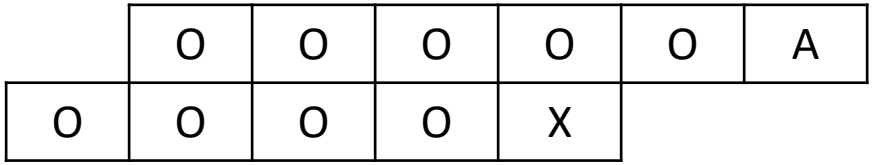
kombinácia dvoch AC – jeden pracuje 20%, druhý 80%

R ≈ 0.2 * 0.881 + 0.8 * 0.384 = 0.483 b/px

Adaptívne aritmetické kódovanie s kontextom

- JBIG (Joint Bi-level Image Experts Group)





- kontext - 10 bitové okolie (O pevné, A sa môže meniť)
- 1024 adaptívnych QM kóderov so škálovaním
- z kontextu spočítame index kódera
- aktualizujeme príslušný QM kóder podľa hodnoty X

Faxové prenosy

- pre ITU-T G3 a G4

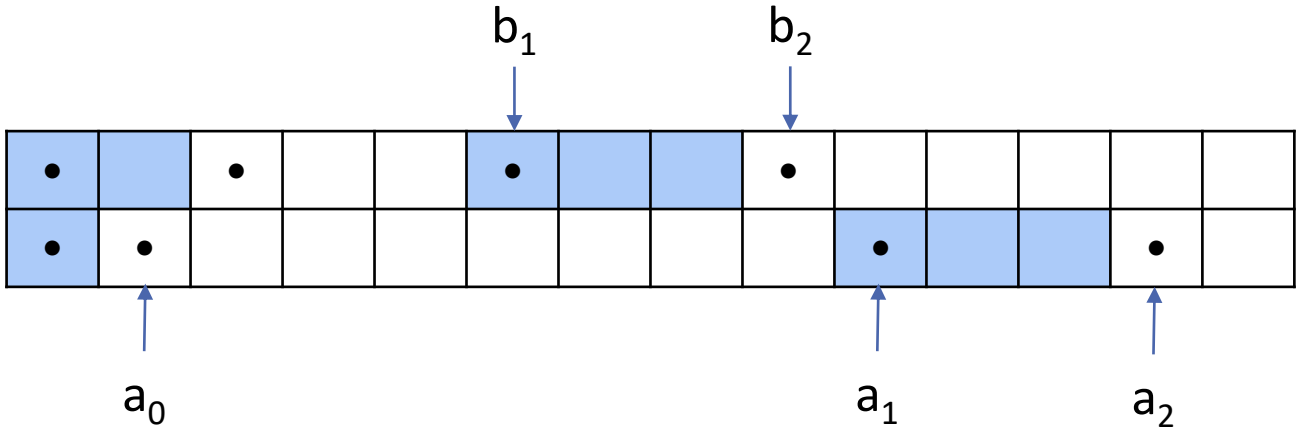
1-dimenzionálne kódovanie RLE

- prvý beh stále biely (môže byť aj 0), potom striedanie sb, sw
- ďalej kódovanie MH – modifikovaným Huffmanovym kódom podľa RG(64) – teda štatistickým kódom pre (s div 64) a štatistickým kódom pre (s mod 64)

2-dimenzionálne vstupy – korelácia medzi riadkami
READ (Relative Element Address Designate)
resp. MR (Modified READ)

- pozície striedania b/w podľa predchádzajúceho riadka
- začiatok – stále (fiktívna) biela

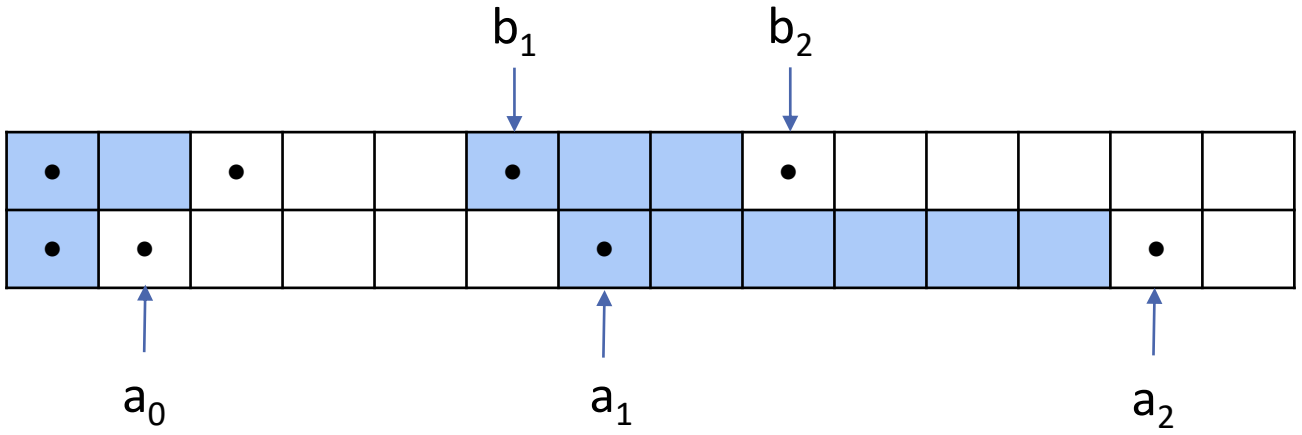
READ (Relative Element Address Designate)



a_0 – posledný pixel, známy príjemcovi (väčšinou pozícia striedania)
a_1 – prvá pozícia striedania za a_0 (opačnej farby)
a_2 – pozícia striedania za a_1 (farby ako a_0) – príjemca nepozná a_1, a_2
b_1 – prvá pozícia striedania v predchádzajúcom riadku, ktorá je napravo od a_0 a má opačnú farbu – pozná aj príjemca
b_2 – ďalšia pozícia striedania v predchádzajúcom riadku

- ak $b_2 < a_1$ pass mode (0001) až po pozíciu pod b_2 bude všetko farby a_0 , nastavíme a_0 na pozíciu pod b_2 a nastavíme aj ostatné

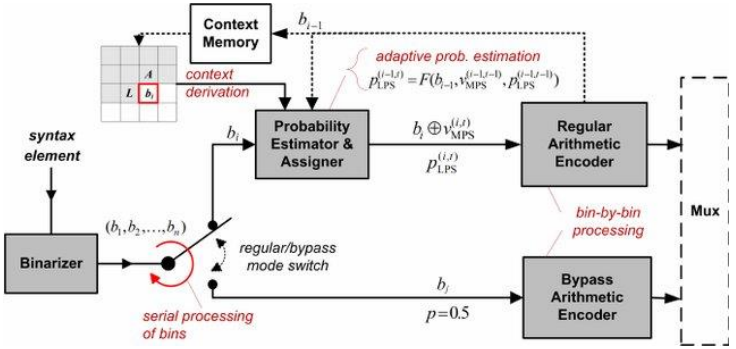
READ (Relative Element Address Designate)



- ak a_1 sa líši od b_1 najviac o 3 – vertical mode – pošleme kódy posunutí (0000010, 000010, 010, 1, 011, 000011, 0000011), nastavíme a_0 na pozíciu a_1 a prepočítame ostatné (b_1,b_2,a_1,a_2)
- inak – horizontal mode (001) a kódujeme sekvenciu od a_0 a od a_1 MH 1D kódom a a_0 nastavíme na a_2
- obmedzenie veľkého počtu 2D kódovaných riadkov – kvôli možným chybám
- verzia MMR (modified modified READ) bez obmedzení

iné bi-level postupy

- JBIG2, halftone, symbol region
- MRC – Mixed Raster Context – background, foreground, mask (text)
- JPEG CALIC – Context Adaptive Losseless Image Compression
- CABAC – Context-based Adaptive Binary Arithmetic Coder (pre H.264/AVC, H.263, MPEG-2, MPEG-4)



JPEG-LS (Joint Photographic Experts Group)

- prediktívne bezstratové kódovanie (losseless) pre JPEG
- kódovanie rozdielom od predikcie
- jednorozmerné p'[i] = p[i-1] c[i]=p[i]-p'[i];
  p[i]=p[i-1]+c[i]
- p''[i] = p[i-1]+(p[i-1]-p[i-2]) c[i]=p[i]-p''[i]
  p[i]= 2p[i-1]-p[i-2]+c[i]
- kódovanie RG kódmi s vhodným m
- namiesto unárneho kódu so znamienkom –
  Huffmanovo kódovanie pre triedu
  0 (-1,1) (-3,-2,2,3) (-7,-6,-5,-4,4,5,6,7) ...
  v rámci triedy – blokový kód

JPEG-LS (Joint Photographic Experts Group)

2D prediktívne kódovanie – rozdiel od predikcie

1. N
2. W
3. NW
4. N+W-NW
5. W+(N-NW)/2
6. N+(W-NW)/2
7. (N+W)/2

NW	N	NE
W	*	

spočíta sa kompresný pomer pre všetky – vyberie najlepší

ppm (prediction with partial match)

- predikcia s čiastočnou zhodou – adaptívny algoritmus
- kontext -1 rádu - všetky používané znaky s váhou 1
- kontext 0 rádu – už použité znaky s váhou = počtom použití
- kontext 1 rádu pre každý použitý znak tabuľka početností všetkých pokračovaní
- kontext 2 rádu – pre každú použitú dvojicu znakov tabuľka početností pokračovaní ... (cca do 5)
- v každej tabuľke symbol ESC – pokračovanie, ktoré ešte nebolo registrované
- aktualizované hodnoty je možné uchovávať v rozhodovacom strome (trie) – rýchly prístup

ppm (prediction with partial match)

- začíname len s kontextom -1
  pre celočíselné AC bude L=0, H=11..1
- kódovanie znaku x začneme nájdením jeho najväčšej kontextovej tabuľky (obidve strany sú v rovnakej situácii)
- ak sa v nej znak x nachádza, spočítame kód podľa tejto tabuľky (podľa napočítaných počtov výskytov a celkovej sumy)
- ak nie, pošlem kód pre ESC a pokračujem s menším kontextom v predchádzajúcom kroku (kontext -1 bude vyhovovať všetkým)
- aktualizujem početnosti v relevantných tabuľkách
- po každom odoslaní kódu – škálovanie ...
- ppma – ESC je stále s početnosťou 1, ppmc – ESC zvyšujeme pri každom odoslaní, ppm* - početnosť ESC podľa počtu položiek v tabuľke

Progresívne postupy

- HINT – hierarchická predikcia – hierarchicky po vrstvách – pre kódovanie CAT (computerized axial tomography), MRI (magnetic resonance images)
- najskôr zakódovať kľúčové body, ostatné postupne v progresívnom rozlíšení pomocou rozdielov od lineárnej predikcie

Bidirectional code

- B – kód od začiatku vstupu
  B´ - kód od konca vstupu v reverznom tvare
- C = 00..0B´ $\oplus$ B00..0
  kde 00..0 je dlhšie ako najdlhšie kódové slovo

Ďakujem za pozornosť.

jozef.jirasek@upjs.sk