1 Range encoding

Spunând că lăţimea unui mediu de stocare este *s*, sau *d* cifre din baza *b*, înţelegem că poate lua una din cele s valori, sau una din cele b^d valori distince.

Dacă stocăm o literă, şi restrângem mediul de stocare la una din t valori distincte, atunci lăţimea codării caracterului este s/t, şi lăţimea rămasă este t, în care putem stoca un REST de lăţime *t* . Setul de t valori ce pot reprezenta litera, se numeşte DOMENIUL literei în lăţimea spaţiului de stocare.

De exemplu dacă domeniul unei litere într-un spaţiu de stocare cu lăţimea 256 este [240, 250), atunci lăţimea lilterei este 25.6, şi lăţimea rămasă este 10.

Dacă un domeniu are forma $[B, T)$, atunci îl putem combina cu un rest prin aritmetică simplă. Dacă dorim să stocăm $i \in [0, T - B)$, ca rest pentru $[B, T)$, atunci valoarea stocată este $B + i$; sau dacă $[i, j) \subseteq [0, T - B)$trebuie stocat ca rest parţial pentru $[B, T)$, atunci valoarea stocată este constrâns la $[B + i, B + j)$.

Fie $f(a)$ probabilitatea ca litera 'a' să apară în orice context dat. Presupunem că alfabetul este ordonat, şi definim $F(a)$ ca fiind probabilitatea unei literi precedente lui 'a' să apară în acelaşi context, adică:

$$F(a) = \sum_{x<a} f(x)$$

În continuare voi nota $f(a)$ cu fa, $F(a)$ cu Fa, $s.fa$ cu sfa.

Shannon a arătat, că pentru minimizarea cifrelor necesare pentru reprezentarea mesajului într-o bază *b*, ar trebui să codăm fiecare literă *'a'*, a.î. lăţimea să fie $-\log_b(fa)$ cifre, adică $1/fa$ în lăţime absolută.

Nu putem realiza acest lucru exact, dar dacă codăm *'a'* într-un spaţiu de stocare cu lăţimea *s*, ca şi $[\lfloor s.Fa \rfloor , \lfloor s(fa + F$ atunci lăţimea literei se aproprie de $1/fa$ pentru $s.fa \gg 1$. Dacă $s.fa \geq 1$, atunci fiecare literă se poate coda, şi decoda fără echivoc.

1.1 Decodificare

O literă *'a'*, împreună cu restul său este codificat (într-un spaţiu de stocare de lăţime *s*) ca $i \subseteq [\lfloor sFa \rfloor , \lfloor s(Fa + fa) \rfloor]$. Fie $L(j)$ ultima literă *e* din alfabet pentru care $Fe < j$. Putem folosi *L* pentru a deduce *'a'*, ştiind *i*:

$$\lfloor sFa \rfloor \;\leq\; i < \lfloor s(Fa + fa) \rfloor \quad \Rightarrow sFa < i + 1 \leq s(Fa + fa) \quad \Rightarrow Fa < \frac{i+1}{s} \leq Fa + fa$$

$\Rightarrow a = L\left(\frac{i+1}{s}\right)$

Trebuie ţinut cont şi de erorile de rotunjire la calcularea lui $\frac{i+1}{s}$. Putem verifica dacă litera este corectă prin confirmarea relaţiei $\lfloor sFa \rfloor \leq i < \lfloor s(Fa + fa) \rfloor$.

După ce am dedus *'a'*, restul este $i - \lfloor sFa \rfloor$, şi a fost codat cu o lăţime de $\lfloor s(Fa + fa) \rfloor - \lfloor sFa \rfloor$.

1.2 Algoritmul de codificare/decodificare

Dacă o literă *'a'* se codifică ca $[B, T)$, lăţimea rămasă este $T - B$. Dacă acesta e prea mic, îl putem extinde prin adăugarea unei cifre (în baza *b*), domeniul devenind: $[Bb, Tb)$, şi lăţimea rămasă devine $(T-B)b$. La decodificare ignorăm cifra în plus, pentru că codificarea lui *'a'* în lăţimea sb nu este neapărat $[Bb, Tb)$.

Fie $s = b^w$, unde w este numărul (întreg) maxim de cifre în baza *b* pe care îl putem utiliza în mod convenabil.

Codificăm prima literă a mesajului în lăţimea *s*, şi adăugăm atâtea cifre în coadă, cât putem fără să cauzăm ca restul să depăşească lăţimea *s*.

Fie lăţimea spaţiului de stocare după codarea a celei de a *i*-a literă: S_i, de valoare $[B_i, T_i)$; atunci putem coda următoarea literă $A(i + 1)$, in spaţiul de stocare de lăţime $R(i + 1)$, unde:

$$\begin{aligned} R_{i+1} &= (T_i - B_i)b^{k(i+1)} \\ k_{i+1} &= w - \lceil \log_b(T_i - B_i) \rceil \end{aligned}$$

Pentru $i > 0$:

$$
\begin{aligned}
[B_i, T_i) &= \left[B_{i-1}b^{k_i} + \lfloor R_i F A_i \rfloor\, , B_{i-1}b^{k_i} + \lfloor R_i (F A_i + f a_i) \rfloor\right) \\
S_i &= \sum_{j=1}^{i} k_j \\
[B_0, T_0) &= [0, 1)
\end{aligned}
$$

1.3 Exemplu de codificare

Codificarea mesajului: *"NMLNNNKKNML"*

Lăţime rămasă (ajustat)	litera următoare	domeniul literei următoare	Mesaj curent codificat	Domeniul curent mesajului	Lăţime rămasă
1000	N	$[580, 1000)$	N	$[580, 1000)$	420
420	M	$[130, 243)$	NM	$[710, 823)$	113
113	L	$[011, 035)$	NML	$[721, 745)$	24
240	N	$[139, 240)$	NMLN	$[7349, \ldots 450)$	101
101	N	$[058, 101)$	NMLNN	$[7407, \ldots 450)$	43
430	N	$[249, 430)$	NMLNNN	$[74319, \ldots 500)$	181
181	K	$[000, 018)$	NMLNNNK	$[74319, \ldots 337)$	18
180	K	$[000, 018)$	NMLNNNKK	$[743190, \ldots 208)$	18
180	N	$[104, 180)$	NMLNNNKKN	$[7432004, \ldots 080)$	76
760	M	$[235, 440)$	NMLNNNKKNM	$[73420275, \ldots 480)$	205
205	L	$[020, 063)$	NMLNNNKKNML	$[73420295, \ldots 338)$	43

Codul complet trebuie ales cu 7 cifre semnificative (din: $[73420295, 73420338)$), de ex: 7432031.

1.4 Implementare algoritm

Se observă că în cazul unui domeniu există 3 zone distincte:

$$
\begin{array}{ll}
[& \underbrace{13}\ \underbrace{19}\ \underbrace{314} \quad , \\
 & \underbrace{13}_{z1}\ \underbrace{20}_{z2}\ \underbrace{105}_{z3} \quad]
\end{array}
$$

Zona z1 constă din cifre comune tuturor numerelor din domeniu, deci nu vor fi afectate de alegerea restului. Aceste cifre pot fi scrise la ieşire.

Zona_z2 constă din *n* cifre formând un număr db^{n-1}, sau $db^{n-1} - 1$, unde *d* este o singură cifră, şi *b* este baza codificării. În aceste exemplu $n = 2$, şi $d = 2$. Cifrele din această zonă pot fi afectate de alegerea restului, dar care nu sunt necesare pentru a distinge 2 numere din domeniu. Acestea le numim cifre AMÂNATE, şi (d, n) identifică posibilele valori ale cifrelor. Prin convenţie, dacă $n = 0 \quad \Rightarrow d = 0$.

Zona z3 constă din *w* cifre, şi sunt suficiente pentru a distinge între 2 numere din domeniu.

Considerăm domeniul $[B', T']$, cu cifrele transmise: c, şi cifrele amânate reprezentate prin (d, n). Fie *x* cifrele transmise după rezolvarea amânării superior:

$$x = cb^n + db^{n-1}$$

atunci putem exprima $[B', T']$, ca: $c, (d, n), [B, T]$, unde $B = B' - xs$, şi $T = T' - xs$. De exemplu $[1319314, 1320105)$ devine $13, (2, 2), [-686, 105)$.

Dacă lăţimea rămasă este $T - B$, şi dacă combinăm $c, (d, n), [B, T]$ cu restul parţial $[i, j) \subseteq [0, T - B]$, atunci creăm domeniul $c, (d, n), [B + i, B + j]$.

Dacă $B + j \leq 0$ atunci putem rezolva cifra amânată inferior, iar dacă $B + i \geq 0$atunci îl putem rezolva superior.

Acest algoritm se poate implementa simplu, fiindcă, dacă domeniul este $c, (d, n), [B, T]$, atunci: $-s < B < T \leq +s$, unde:

d este o singură cifră

n este un întreg mic

c nu trebuie reţinut în codificator/decodificator

Pentru a limita numărul de cifre amânate, putem impune o limită superioară. Putem forţa rezolvarea amânării prin modificarea capetelor domeniului.

Ex:

$$13, (2, 3), [-660, 140] \Rightarrow 13, (2, 3), [-660, 000] \Rightarrow 13199, (0, 0), [340, 1000)$$

$$13, (2, 3), [-140, 660) \Rightarrow 13, (2, 3), [000, 660) \Rightarrow 13200, (0, 0), [000, 660)$$

Prin acesta risipim cel mult 1 bit.

Bibliografie

[1] G.N.N Martin - "Range encoding: an algorithm for removing redundancy from a digitised message.", Video & Data Recording Conference, Southampton, 1979, http://www.compressconsult.com/rangecoder/rngcod.pdf.gz

[2] Alan Silverstein - "Judy IV Shop Manual", HP invent, August 2002, http://judy.sourceforge.net/doc/shop_interm.pdf

[3] Peter Deutsch - "DEFLATE Compressed Data Format Specification version 1.3 (RFC 1951)", May 1996 - ftp://ftp.nic.it/rfc/rfc1951.pdf

[4] Matthew V. Mahoney - "Adaptive Weighing of Context Models for Lossless Data Compression", Florida Institute of Technology CS Dept, Technical Report CS-2005-16, https://www.cs.fit.edu/Projects/tech_reports/cs-2005-16.pdf