1. mājasdarbs

Lietišķie algoritmi: Bezzudumu saspiešana
Termiņš: 2020-10-05

Atrisinājumus lūdzam pārveidot par vienu PDF.

Vairāki uzdevumi šajā mājasdarbā iespaidojušies no MIT Open Courseware: `https://bit.ly/37Aywtf` un `https://bit.ly/2BdK8GA`

1.uzdevums (Gabalu garumu kodējums).
Aplūkojam *Bernulli gadījumlielumu* X, kam

$$\begin{cases} P(X=0) = \dfrac{255}{256}, \\ P(X=1) = \dfrac{1}{256}. \end{cases}$$

(Piemēram, tiek mesta asimetriska monēta, kam "cipars" (*heads* jeb bits 1) uzkrīt ar varbūtību $\frac{1}{256}$.)
Virknīti ar n šī gadījumlieluma vērtībām apzīmē ar X^n. To saspiež, izmantojot *Gabalu garumu kodējumu* (*run-length encoding*). Katru vieninieku kodē atsevišķi, bet nepārtrauktiem nuļļu gabaliem izvada tikai gabalu garumus.
Algoritma apraksts:
Ievadei X^n saspiesto rezultātu $f(X^n)$ iegūst, atkārtoti izpildot šos 3 soļus, ko turpina līdzkamēr ievade pilnībā nolasīta.

1.Solis
Ja ievadē pirmais bits ir "1", tad to nolasa un izvadē raksta astoņas nulles: `00000000`.

2.Solis
Ja ievades sākumā ir $r \leq 255$ nulles, aiz kurām seko vieninieks, tad visas šīs nulles nolasa un izvadē raksta 8-bitu virknīti – skaitļa r bināro pierakstu. (Tā kā nuļļu skaits $r \neq 0$, tad šajā solī nerodas izvade `00000000`.)

3.Solis
Ja ievadē ir $r > 255$ pēc kārtas esošas nulles, tad šajā solī nolasa tikai 255 nulles (un izvadē raksta skaitli 255 bināri, kā iepriekšējā punktā: `11111111`). Turpina pildīt 3.soli (un izvadīt skaitļa 255 kodus) līdzkamēr iestājas 1. vai 2. soļa situācija.

Atrast saspiešanas attiecības robežu šim kodējumam:

$$\lim_{n\to\infty} \frac{1}{n} E(\ell(f(X^n)). \qquad (1)$$

Ar E apzīmējam vidējo vērtību gadījumlielumam, bet $\ell(f(X^n))$ apzīmē saspiestās virknītes garumu.
(Šī ir parodija par 4.uzdevumu no `https://bit.ly/2Y6mUKa`.)

2.uzdevums (Gadījuma analīze).
Aplūkojam ziņojumu alfabētu ar četriem ziņojumiem: $S = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$, kuru varbūtības ir attiecīgi $\{1/3, 1/3, 2/9, 1/9\}$.

1. Izmantot Hafmana algoritmu, lai atrastu šim ziņojumu alfabētam optimālu bezprefiksu kodējumu.
2. Vai ar Hafmana algoritmu var konstruēt būtiski citādu optimālu bezprefiksu kodējumu (t.i. tādu, kas simboliem `a`, `b`, `c`, `d` piekārto citādus kodu garumus, nevis tikai kaut kur aizstāj `0` ar `1` un otrādi).
3. Atrast bezprefiksu kodējumu, kurš arī ir optimāls, bet nevar būt radies Hafmana algoritma rezultātā.

(Šis ir uzdevums 2.12. no `https://bit.ly/2Y4PKMe`, 55.lpp.)

(A) Saspiežot ar aritmētisko kodu, saspiestās virknes garums ir ziņojumu virknes garums reiz entropija $H(S)$. Mūsu gadījumā, risinot Markova ķēdes stabilam stāvoklim atbilstošo lineāro vienādojumu sistēmu, iegūstam, ka burta "a" varbūtība ir $1/2$, "b" un "c" varbūtības ir $1/4$. Tādēļ entropija $H(S) = 1.5$ (kods pagarinās par vienu bitu, ja iekodē "a", un par diviem bitiem, ja iekodē "b" vai "c"); vidēji vienam burtam tērējas 1.5 biti.
Sākotnējās virknes garums ir 8000000 biti (jeb 1000000 baiti). Pēc saspiešanas tie ir $1.5 \cdot 1000000 = 1500000$ biti (jeb 187500 baiti). Attiecība saspiestās virknes garumam baitos pret sākotnējā faila garumu (baitos) ir 0.1875. Saspiestais fails aizņem 18.75% no nesaspiesta faila izmēra.
Piezīme. Ja Jūsu pieņēmumi par sākotnējā faila kodējumu atšķīrās, tad saspiešanas attiecība var iznākt arī cita. Nav grūti "a","b","c" kodēt pat ar diviem baitiem (daži Unikoda kodējumi tā dara) vai arī ļoti taupīgi (teiksim, ar 2 bitiem katru burtu, jo burtu ir tikai 3.)
(B) Saspiežot ar WinZip (šādu eksperimentu apraksta Aleksejs Jelisejevs) iznāk 115050 baiti. Citos mājasdarbos ir arī citi dati 132000 baiti utml. Kā redzams, vismaz dažos gadījumos WinZip (kas izmanto Lempela-Ziva saimes algoritmus) var saspiest failus labāk par entropiju.
(C) Hafmana kodam (ņemot vērā to, ka burtu a,b,c varbūtības ir $1/2, 1/4, 1/4$ arī izdodas sasniegt entropiju. T.i. saspiešanas attiecība ir 18.75% (tāpat kā (A) piemērā).
(D) Burtu pārīšu varbūtības (iegūstam, apskatot veidus, kā tos iegūt no automāta stāvokļiem):
`aa` $\rightarrow 1/4$,
`ab` $\rightarrow 1/4$,
`ba` $\rightarrow 1/8$,
`bc` $\rightarrow 1/8$,
`ca` $\rightarrow 1/8$,
`cc` $\rightarrow 1/8$.
Entropija šiem sešiem ziņojumiem ir 2.5 (starp 2 un 3 biti uz vienu pārīti) jeb 1.25 uz vienu burtu. Tātad saspiešanas attiecības robeža būs $1.25 \cdot 1000000/8000000 = 0.156250$. Redzam, ka simbolu pārīšiem entropijas kodi (Hafmans vai aritmētiskais) saspiež labāk nekā atsevišķiem simboliem. No otras puses, līdz WinZip līmenim šādi tik ir grūti.

□

3.uzdevums (Markova process).
Pieņemsim, ka *Markova process* (Attēls 1) ir jau sasniedzis stabilu stāvokli (jau ir notikusi pietiekami gara nejauša staigāšana pa šo varbūtisko grafu, lai simbolu proporcijas un secību vairs nenoteiktu sākumstāvoklis).
Aplūkojam stabilā Markova procesa ģenerētu simbolu virkni X^n, kur X pieņem vērtības no $\{\texttt{a}, \texttt{b}, \texttt{c}\}$.

1. Atrast saspiešanas attiecības robežu, ja simbolu virkni X^n saspiež ar aritmētisko kodu.

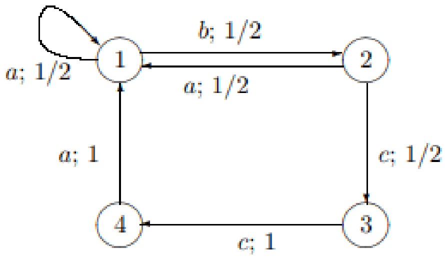

Figure 1: Markova process.

2. Ar datorsimulāciju izveidot virkni X^n garumā $n = 10^6$. Atrast saspiešanas attiecību, ja izmanto WinZip arhivatoru (WinZip parasti lieto *DEFLATE* algoritmu, kas ir modificēts LZ77 un vēl arī Hafmana kods.)

3. Atrast saspiešanas attiecības robežu ($n \to \infty$), ja Markova virkni X^n saspiež ar Hafmana kodu, piešķirot prefiksu kodus atsevišķiem burtiem $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$.

4. Atrast saspiešanas attiecības robežu ($n \to \infty$), ja Markova virkni X^n saspiež ar Hafmana kodu, piešķirot prefiksu kodus burtu pāriem

$$\{\mathtt{aa}, \mathtt{ab}, \ldots, \mathtt{cc}\}.$$

(Neiespējamie burtu pāri prefiksu kokā nav jāattēlo.)

(Šī ir parodija par 2.30.uzdevumu no `https://bit.ly/2Y4PKMe`, 60.lpp.)

4.uzdevums (LZ78 apakšējais novērtējums).
Bobs vēlas izveidot bitu virknīti jeb stringu, uz kura LZ78 algoritms uzvedas vissliktāk (virknes garums pēc arhivēšanas sanāk visgarākais). Šai nolūkā viņš izraksta visas netukšās galīgās bitu virknītes *Shortlex order* secībā: vispirms leksikogrāfiski sakārtotas viena bita virknītes, tad leksikogrāfiski sakārtotas divu bitu virknītes utml. Rezultātā viņam iznāk sekojoša virkne:

0.1.00.01.10.11.000.001.010.011.100.101.110.111 ...

Bobs šo virkni pārstāj rakstīt tad, kad ir izrakstītas visas bitu virknītes līdz garumam k ieskaitot. (Piemērā dota virknīte tad, ja $k = 3$.) Punkti ievietoti tikai uzskatāmības dēļ; LZ78 ievade faktiski satur tikai simbolus "0" un "1".
Boba uzrakstīto bitu stringu (bez atdalītājpunktiem) apzīmēsim ar W_k. To arī izmantojam par LZ78 algoritma "sliktāko ievadi", kas cenšas panākt, lai Lempels-Zivs retāk izmantotu virknes no vārdnīcas, lai kods sanāktu iespējami garš.

1. Pierādīt, ka Boba uzrakstītās virknītes garums $|W_k| = (k-1)2^{k+1} + 2$.

2. Ar $c(W_k)$ apzīmējam to kodu skaitu, kas tiek ievietoti vārdnīcā, ja iekodē W_k. Atrast, ar ko vienāds $c(W_k)$ (formula, kas atkarīga no k).

3. Uzrakstīt LZ78 kodu, kas rodas iekodējot W_3 (sk. slaidu "LZ78 (biti par bitiem)"). Vai ir iespējams, ka LZ78 kods (bitu virkne) ir garāks par ievadē saņemto bitu virkni?

Piezīme: Šis uzdevums izmanto LZ78 variantu, kas sāk ar tukšu vārdnīcu. Piemēru sk. `https://bit.ly/3i7HEsU`. (Šo nevajag jaukt ar LZW algoritmu, kurš ir līdzīgs, bet sākumā ir jau sabāzis vārdnīcā visu ziņojumu alfabētu - katram alfabēta burtam ir jau unikāls kods.)

5.uzdevums (I-iespēja).
Teksts satur $n = 2^k$ dažādus simbolus:

$$S = \{s_1, s_2, \ldots, s_n\}.$$

To varbūtības izkārtotas dilstošā secībā:

$$p_1 \geq \ldots \geq p_n.$$

Zināms, ka ziņojumu kopai S optimālu prefiksu kodējumu veido visas 2^k vienāda garuma k-bitu virknītes.

1. Vai noteikti ir spēkā apgalvojums: Populārākā ziņojuma varbūtība apmierina nevienādību:
$$p_1 \leq \frac{2}{2^k + 1}?$$

2. Vai noteikti ir spēkā apgalvojums: Populārākā ziņojuma varbūtība nepārsniedz divu retāko ziņojumu varbūtību summu: $p_1 \leq p_{n-1} + p_n$?

3. Vai noteikti ir spēkā apgalvojums: Populārākā ziņojuma varbūtība nepārsniedz divkāršotu visretākā ziņojuma varbūtību: $p_1 \leq 2p_n$?

(Šī ir parodija par 16.3-8.uzdevumu no *Introduction to Algorithms. Third Edition.* Cormen, Leiserson, Rivest, 2009.)