Vidus eksāmens

Lietišķie algoritmi, 2019.g. rudens
Termiņš: 2019-10-29

1.uzdevums (1+1+1+2 punkti): Spēlētājs X vienā gājienā izņem no urnas trīs kartiņas. Pieņemsim, ka urna ir ļoti liela, kartiņas tajā nekad nebeidzas un ir vienādas varbūtības izķeksēt jebkuru no burtiem `A`, `B` vai `C`; citu burtu urnā nav. Pēc tam X sakārto trīs kartiņas alfabētiskā secībā un nosūta spēlētājam Y ziņojumu – to burtu, kurš pēc sakārtošanas bija pirmais. (Piemēram, ja izķeksētie burti ir `"CBC"`, tad pēc sakārtošanas tie būs `"BCC"` un X nosūta ziņojumu `"B"`.)

1. Kāds ir informācijas saturs ziņojumam `A`?
2. Kāds ir informācijas saturs ziņojumam `B`?
3. Kāds ir informācijas saturs ziņojumam `C`?
4. Kāda ir entropija jebkuram vienam ziņojumam, ko X nosūta Y saskaņā ar augšminēto procedūru?

1. No visiem 27 variantiem, kā izvilkt kartiņas, ir 9 tādi, kas sākas ar burtu `A`. Starp atlikušajiem 18 trešajai daļai (jeb 6 variantiem) `A` ir otrajā pozīcijā. Visbeidzot, starp atlikušajiem $27 - 9 - 6 = 12$ ir trešdaļa jeb 4 tādi, kam `A` ir trešajā pozīcijā. Tātad ziņojuma `A` varbūtība ir $(9 + 6 + 4)/(27) = 19/27$.
   Informācijas saturs: $-\log_2(P(\mathtt{A})) \approx 0.507$.
2. Ziņojuma `B` varbūtība ir $7/27$ (no 1 atņem ziņojumu `A` un `C` varbūtības).
   Informācijas saturs ir $-\log_2(P(\mathtt{B})) \approx 1.948$.
3. Ziņojums `C` var rasties vienīgi izvelkot kartiņas `CCC`, tā varbūtība ir $p(\mathtt{C}) = 1/27$.
   Informācijas saturs ir $-\log_2(P(\mathtt{C})) \approx 4.754$.
4. Entropija ir

$$-P(\mathtt{A})\log_2(P(\mathtt{A})) - P(\mathtt{B})\log_2(P(\mathtt{B})) - P(\mathtt{C})\log_2(P(\mathtt{C})) =$$

$$= (19/27)\cdot 0.507 + (7/27)\cdot 1.948 + (1/27)\cdot 4.755 \approx 1.038.$$

Piezīme. Variantu skaitu, kuros ir vismaz viens burts `A` var saskaitīt arī citādi. Ar U apzīmējam visus tos variantus, kuros pirmais burts ir `"A"`, ar V – tos, kuros otrais burts ir `"A"`, un ar W – tos, kuros trešais burts ir `"A"`; sk. Attēlu 1. Viegli redzēt, ka elementu skaits kopās $|U| = |V| = |W| = 9$, elementu skaits kopu šķēlumos ir $|U \cap V| = |U \cap W| = |V \cap W| = 3$ (ir 3 tādi varianti, kuros pirmais **un** otrais burts ir `"A"`). Savukārt visu trīs kopu šķēlums $|U \cap V \cap W| = 1$ (atbilst gadījumam, kad visi trīs burti ir `"A"`).
Sk. `https://bit.ly/375cX4L` (ieslēgšanas-izslēgšanas principu), kas ļauj noteikt elementu skaitu visu šo trīs kopu apvienojumā:

$$|U \cup V \cup W| = |U| + |V| + |W| - |U \cap V| - |U \cap W| - |V \cap W| + |U \cap V \cap W|\,.$$

$$|U \cup V \cup W| = 9 + 9 + 9 - 3 - 3 - 3 + 1 = 19.$$

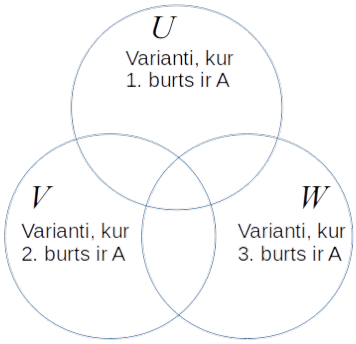


Figure 1: Kopu apvienojuma izteikšana ar ieslēgšanas-izslēgšanas principu

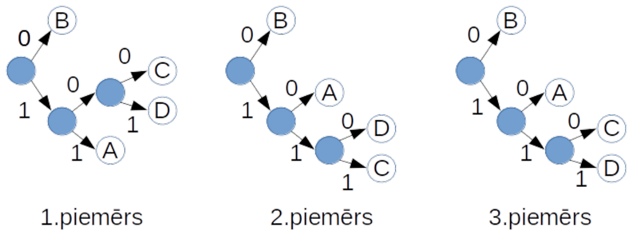
Citiem vārdiem, lai atrastu, cik elementu pieder kaut vienai no kopām U, V, W, saskaitām šajās kopās esošos elementus ($|U|$, $|V|$ un $|W|$), tad atņemam tos, kurus esam pieskaitījuši divreiz (kuri pieder kādam no divu kopu šķēlumiem $|U \cap V|$, $|U \cap W|$ un $|V \cap W|$), visbeidzot pieskaitām atpakaļ tos, kuri pieder visām trim ($|U \cap V \cap W|$).

$\square$

2.uzdevums (3 punkti - jebkāds pareizs Hafmana koks, 3 punkti - koks kanoniskajā formā): Hafmana koku sauksim par *kanonisku*, ja izpildās sekojošas īpašības:

- Visi kanoniskā koka zari (ceļi no saknes līdz lapām/ziņojumiem) veido garumus, kuri ir nedilstošā secībā, skaitot no augšas uz leju.
- Vienāda garuma zariem ziņojumi izkārtoti ziņojumu alfabētiskā secībā.

Attēla 1.piemērā zaru garumi nav nedilstošā secībā (zars `11` uz lapu `A` ir garumā 2, virs tā divi zari garumā 3). 2.piemērā `C`, `D` ir ar vienādi gariem kodavārdiem, bet nav alfabētiskā secībā. Vienīgi 3.piemērā Hafmana koks ir kanonisks.



5 ziņojumu kopai $\{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}\}$, kuru varbūtības ir attiecīgi $\left\{\frac{1}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}, \frac{5}{15}\right\}$, atrast un uzzīmēt kanonisku Hafmana koku.

Vispirms veidojam jebkādu Hafmana koku (kaut vai nekanonisku). Sākumā mums ir 5 mini-koki, šos kokus pierakstām kā pārīšus, piemēram, $(\mathtt{A}, 1/15)$, kur ziņojums sapārots ar savu svaru. Katrā nākamajā solī apvienojam divus vieglākos kokus lielākā kokā (koku, ko veido apakškoki t_1 un t_2 apzīmējam ar $T(t_1, t_2)$) un saskaitām to svarus.

$$
\begin{aligned}
& \left\{\left(\mathtt{A}, \frac{1}{15}\right), \left(\mathtt{B}, \frac{2}{15}\right), \left(\mathtt{C}, \frac{3}{15}\right), \left(\mathtt{D}, \frac{4}{15}\right), \left(\mathtt{E}, \frac{5}{15}\right)\right\} && (1)\\
\to\ & \left\{\left(T(\mathtt{A},\mathtt{B}), \frac{3}{15}\right), \left(\mathtt{C}, \frac{3}{15}\right), \left(\mathtt{D}, \frac{4}{15}\right), \left(\mathtt{E}, \frac{5}{15}\right)\right\} && (2)\\
\to\ & \left\{\left(\mathtt{D}, \frac{4}{15}\right), \left(\mathtt{E}, \frac{5}{15}\right), \left(T(T(\mathtt{A},\mathtt{B}),\mathtt{C}), \frac{6}{15}\right)\right\} && (3)\\
\to\ & \left\{\left(T(T(\mathtt{A},\mathtt{B}),\mathtt{C}), \frac{6}{15}\right), \left(T(\mathtt{D},\mathtt{E}), \frac{9}{15}\right)\right\} && (4)\\
\to\ & \{(T(T(T(\mathtt{A},\mathtt{B}),\mathtt{C}),T(\mathtt{D},\mathtt{E})),1)\}\,. && (5)
\end{aligned}
$$

Uzzīmēsim (nekanonisko) Hafmana koku un pārveidosim to par kanonisku, sakārtojot zarus pēc garuma, un vienāda garuma zariem sarakstot ziņojuma burtus alfabētiskā secībā, bet neizmainot ziņojumiem piešķirto Hafmana kodavārda garumus (sk. Attēlu 2). Tādējādi jaunizveidotais, kanoniskais Hafmana koks ir ekvivalents sākotnējam (saglabā koda garumus visiem ziņojumiem), bet tam ir paredzamāka struktūra un šo koku var iekodēt ar mazāk bitiem.

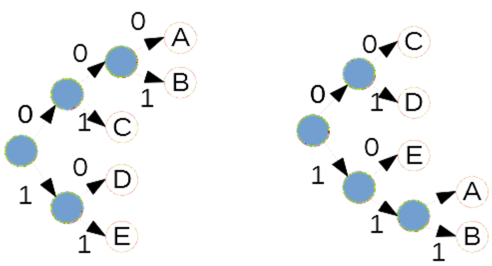


Figure 2: Koka pārveidojums par kanonisku

□

3.uzdevums (4 punkti): Izmantojot LZ78 algoritmu 6 simbolu alfabētam $\{\mathtt{C},\mathtt{E},\mathtt{L},\mathtt{N},\mathtt{S},\mathtt{U}\}$, izveidot tabulu un nokodēt sekojošu 15 simbolu ziņojumu: `SUCCESSLESSNESS`. Tabulā attēlot soļa numuru, w - garāko vārdnīcā jau atrodamo simbolu virkni, k - virknei w sekojošo simbolu, algoritma izvadi un vārdnīcai attiecīgajā solī pievienojamo vārdu.

Solis	w	k	Izvade	Pievieno vārdnīcai
...	...	...	...	...

Solis	w	k	Izvade	Pievieno vārdnīcai
1	S	U	S	SU
2	U	C	U	UC
3	C	C	C	CC
4	C	E	C	CE
5	E	S	E	ES
6	S	S	S	SS
7	S	L	S	SL
8	L	E	L	LE
9	ES	S	ES $\to$ 5	ESS
10	S	N	S	SN
11	N	E	N	NE
12	ESS	[EOF]	ESS $\to$ 9	–

Nokodētais ziņojums ir `S.U.C.C.E.S.S.L.5.S.N.9`.

$\square$

4.uzdevums (6 punkti): Veikt inverso Berouza-Vīlera transformāciju 17 simbolu virknītei: `SETTNRSDDIESIEESN`, ja zināms, ka kodētais vārds ir pats pirmais starp leksikogrāfiski sakārtotajām cikliskajām permutācijām.

Berouza-Vīlera atkodēšanu veicam vairākos soļos: Vienā solī pierakstām priekšā matricai kolonnu, kas sakrīt ar atkodējamo virkni (solis `add`), pēc tam sakārtojam rindiņas alfabētiski (solis `sort`). Minēto soli atkārto tik daudz reižu, cik simbolu ir jāatkodē.

$$\xrightarrow{\text{add}} \begin{pmatrix} \texttt{S}\\ \texttt{E}\\ \texttt{T}\\ \texttt{T}\\ \texttt{N}\\ \texttt{R}\\ \texttt{S}\\ \texttt{D}\\ \texttt{D}\\ \texttt{I}\\ \texttt{E}\\ \texttt{S}\\ \texttt{I}\\ \texttt{E}\\ \texttt{E}\\ \texttt{S}\\ \texttt{N} \end{pmatrix} \xrightarrow{\text{sort}} \begin{pmatrix} \texttt{D}\\ \texttt{D}\\ \texttt{E}\\ \texttt{E}\\ \texttt{E}\\ \texttt{E}\\ \texttt{I}\\ \texttt{I}\\ \texttt{N}\\ \texttt{N}\\ \texttt{R}\\ \texttt{S}\\ \texttt{S}\\ \texttt{S}\\ \texttt{S}\\ \texttt{T}\\ \texttt{T} \end{pmatrix} \xrightarrow{\text{add}} \begin{pmatrix} \texttt{SD}\\ \texttt{ED}\\ \texttt{TE}\\ \texttt{TE}\\ \texttt{NE}\\ \texttt{RE}\\ \texttt{SI}\\ \texttt{DI}\\ \texttt{DN}\\ \texttt{IN}\\ \texttt{ER}\\ \texttt{SS}\\ \texttt{IS}\\ \texttt{ES}\\ \texttt{ES}\\ \texttt{ST}\\ \texttt{NT} \end{pmatrix} \xrightarrow{\text{sort}} \begin{pmatrix} \texttt{DI}\\ \texttt{DN}\\ \texttt{ED}\\ \texttt{ER}\\ \texttt{ES}\\ \texttt{ES}\\ \texttt{IN}\\ \texttt{IS}\\ \texttt{NE}\\ \texttt{NT}\\ \texttt{RE}\\ \texttt{SD}\\ \texttt{SI}\\ \texttt{SS}\\ \texttt{ST}\\ \texttt{TE}\\ \texttt{TE} \end{pmatrix} \xrightarrow{\text{add}} \begin{pmatrix} \texttt{SDI}\\ \texttt{EDN}\\ \texttt{TED}\\ \texttt{TER}\\ \texttt{NES}\\ \texttt{RES}\\ \texttt{SIN}\\ \texttt{DIS}\\ \texttt{DNE}\\ \texttt{INT}\\ \texttt{ERE}\\ \texttt{SSD}\\ \texttt{ISI}\\ \texttt{ESS}\\ \texttt{EST}\\ \texttt{STE}\\ \texttt{NTE} \end{pmatrix} \xrightarrow{\text{sort}} \begin{pmatrix} \texttt{DIS}\\ \texttt{DNE}\\ \texttt{EDN}\\ \texttt{ERE}\\ \texttt{ESS}\\ \texttt{EST}\\ \texttt{INT}\\ \texttt{ISI}\\ \texttt{NES}\\ \texttt{NTE}\\ \texttt{RES}\\ \texttt{SDI}\\ \texttt{SIN}\\ \texttt{SSD}\\ \texttt{STE}\\ \texttt{TED}\\ \texttt{TER} \end{pmatrix} \xrightarrow{\text{add}} \begin{pmatrix} \texttt{SDIS}\\ \texttt{EDNE}\\ \texttt{TEDN}\\ \texttt{TERE}\\ \texttt{NESS}\\ \texttt{REST}\\ \texttt{SINT}\\ \texttt{DISI}\\ \texttt{DNES}\\ \texttt{INTE}\\ \texttt{ERES}\\ \texttt{SSDI}\\ \texttt{ISIN}\\ \texttt{ESSD}\\ \texttt{ESTE}\\ \texttt{STED}\\ \texttt{NTER} \end{pmatrix} \xrightarrow{\text{sort}} \begin{pmatrix} \texttt{DISI}\\ \texttt{DNES}\\ \texttt{EDNE}\\ \texttt{ERES}\\ \texttt{ESSD}\\ \texttt{ESTE}\\ \texttt{INTE}\\ \texttt{ISIN}\\ \texttt{NESS}\\ \texttt{NTER}\\ \texttt{REST}\\ \texttt{SDIS}\\ \texttt{SINT}\\ \texttt{SSDI}\\ \texttt{STED}\\ \texttt{TEDN}\\ \texttt{TERE} \end{pmatrix}$$

$$\xrightarrow{\text{add}} \begin{pmatrix} \texttt{SDISI}\\ \texttt{EDNES}\\ \texttt{TEDNE}\\ \texttt{TERES}\\ \texttt{NESSD}\\ \texttt{RESTE}\\ \texttt{SINTE}\\ \texttt{DISIN}\\ \texttt{DNESS}\\ \texttt{INTER}\\ \texttt{EREST}\\ \texttt{SSDIS}\\ \texttt{ISINT}\\ \texttt{ESSDI}\\ \texttt{ESTED}\\ \texttt{STEDN}\\ \texttt{NTERE} \end{pmatrix} \xrightarrow{\text{sort}} \begin{pmatrix} \texttt{DISIN}\\ \texttt{DNESS}\\ \texttt{EDNES}\\ \texttt{EREST}\\ \texttt{ESSDI}\\ \texttt{ESTED}\\ \texttt{INTER}\\ \texttt{ISINT}\\ \texttt{NESSD}\\ \texttt{NTERE}\\ \texttt{RESTE}\\ \texttt{SDISI}\\ \texttt{SINTE}\\ \texttt{SSDIS}\\ \texttt{STEDN}\\ \texttt{TEDNE}\\ \texttt{TERES} \end{pmatrix} \xrightarrow{\text{add}} \begin{pmatrix} \texttt{SDISIN}\\ \texttt{EDNESS}\\ \texttt{TEDNES}\\ \texttt{TEREST}\\ \texttt{NESSDI}\\ \texttt{RESTED}\\ \texttt{SINTER}\\ \texttt{DISINT}\\ \texttt{DNESSD}\\ \texttt{INTERE}\\ \texttt{ERESTE}\\ \texttt{SSDISI}\\ \texttt{ISINTE}\\ \texttt{ESSDIS}\\ \texttt{ESTEDN}\\ \texttt{STEDNE}\\ \texttt{NTERES} \end{pmatrix} \xrightarrow{\text{sort}} \begin{pmatrix} \texttt{DISINT}\\ \texttt{DNESSD}\\ \texttt{EDNESS}\\ \texttt{ERESTE}\\ \texttt{ESSDIS}\\ \texttt{ESTEDN}\\ \texttt{INTERE}\\ \texttt{NESSDI}\\ \texttt{NTERES}\\ \texttt{ISINTE}\\ \texttt{RESTED}\\ \texttt{SDISIN}\\ \texttt{SINTER}\\ \texttt{SSDISI}\\ \texttt{STEDNE}\\ \texttt{TEDNES}\\ \texttt{TEREST} \end{pmatrix} \xrightarrow{\text{add}} \begin{pmatrix} \texttt{SDISINT}\\ \texttt{EDNESSD}\\ \texttt{TEDNESS}\\ \texttt{TERESTE}\\ \texttt{NESSDIS}\\ \texttt{RESTEDN}\\ \texttt{SINTERE}\\ \texttt{DISINTE}\\ \texttt{DNESSDI}\\ \texttt{INTERES}\\ \texttt{ERESTED}\\ \texttt{SSDISIN}\\ \texttt{ISINTER}\\ \texttt{ESSDISI}\\ \texttt{ESTEDNE}\\ \texttt{STEDNES}\\ \texttt{NTEREST} \end{pmatrix} \xrightarrow{\text{sort}} \begin{pmatrix} \texttt{DISINTE}\\ \texttt{DNESSDI}\\ \texttt{EDNESSD}\\ \texttt{ERESTED}\\ \texttt{ESSDISI}\\ \texttt{ESTEDNE}\\ \texttt{INTERES}\\ \texttt{ISINTER}\\ \texttt{NESSDIS}\\ \texttt{NTEREST}\\ \texttt{RESTEDN}\\ \texttt{SDISINT}\\ \texttt{SINTERE}\\ \texttt{SSDISIN}\\ \texttt{STEDNES}\\ \texttt{TEDNESS}\\ \texttt{TERESTE} \end{pmatrix}$$

$$
\xrightarrow{\text{add}}
\begin{pmatrix}
\texttt{SDISINTE} \\ \texttt{EDNESSDI} \\ \texttt{TEDNESSD} \\ \texttt{TERESTED} \\ \texttt{NESSDISI} \\ \texttt{RESTEDNE} \\ \texttt{SINTERES} \\ \texttt{DISINTER} \\ \texttt{DNESSDIS} \\ \texttt{INTEREST} \\ \texttt{ERESTEDN} \\ \texttt{SSDISINT} \\ \texttt{ISINTERE} \\ \texttt{ESSDISIN} \\ \texttt{ESTEDNES} \\ \texttt{STEDNESS} \\ \texttt{NTERESTE}
\end{pmatrix}
\xrightarrow{\text{sort}}
\begin{pmatrix}
\texttt{DISINTER} \\ \texttt{DNESSDIS} \\ \texttt{EDNESSDI} \\ \texttt{ERESTEDN} \\ \texttt{ESSDISIN} \\ \texttt{ESTEDNES} \\ \texttt{INTEREST} \\ \texttt{ISINTERE} \\ \texttt{NESSDISI} \\ \texttt{NTERESTE} \\ \texttt{RESTEDNE} \\ \texttt{SDISINTE} \\ \texttt{SINTERES} \\ \texttt{SSDISINT} \\ \texttt{STEDNESS} \\ \texttt{TEDNESSD} \\ \texttt{TERESTED}
\end{pmatrix}
\xrightarrow{\text{add}}
\begin{pmatrix}
\texttt{SDISINTER} \\ \texttt{EDNESSDIS} \\ \texttt{TEDNESSDI} \\ \texttt{TERESTEDN} \\ \texttt{NESSDISIN} \\ \texttt{RESTEDNES} \\ \texttt{SINTEREST} \\ \texttt{DISINTERE} \\ \texttt{DNESSDISI} \\ \texttt{INTERESTE} \\ \texttt{ERESTEDNE} \\ \texttt{SSDISINTE} \\ \texttt{ISINTERES} \\ \texttt{ESSDISINT} \\ \texttt{ESTEDNESS} \\ \texttt{STEDNESSD} \\ \texttt{NTERESTED}
\end{pmatrix}
\xrightarrow{\text{sort}}
\begin{pmatrix}
\texttt{DISINTERE} \\ \texttt{DNESSDISI} \\ \texttt{EDNESSDIS} \\ \texttt{ERESTEDNE} \\ \texttt{ESSDISINT} \\ \texttt{ESTEDNESS} \\ \texttt{INTERESTE} \\ \texttt{ISINTERES} \\ \texttt{NESSDISIN} \\ \texttt{NTERESTED} \\ \texttt{RESTEDNES} \\ \texttt{SDISINTER} \\ \texttt{SINTEREST} \\ \texttt{SSDISINTE} \\ \texttt{STEDNESSD} \\ \texttt{TEDNESSDI} \\ \texttt{TERESTEDN}
\end{pmatrix}
$$

Atkodēšanas procedūru varētu turpināt, bet ievērojam, ka šajā brīdī jau atkodēti 9 simboli no 17, turklāt atkodējamais vārds ir leksikogrāfiski pirmais. Tādēļ arī, pierakstot atlikušos simbolus, tas sāksies ar prefiksu `DISINTERE`. Ievērojam arī, ka pirms burtiem `DIS` cikliskajās permutācijās vienmēr parādās burti `SS`. Tātad atkodējamā virkne beidzas ar `SS`. Vienīgā šāda virkne ir sestajā rindiņā. Abas virknes sarakstot kopā, iegūstam meklēto vārdu:

$$\overbrace{\texttt{DISINTERE}}\hspace{-0.55em}\underbrace{\hspace{0.55em}\texttt{STEDNESS}}$$

Abas virknes ir garumā 9, bet atkodējamais vārds ir 17 zīmes garš, tādēļ vidējais burts "E" ir kopīgs abām virknēm.

□

5.uzdevums (2+3 punkti): Kvantizācijas algoritms saņem ieejā attēla krāsu intensitātes, kuras ir veseli skaitļi $x \in \{0, 1, 2, \ldots, 255\}$ un atgriež apakšējo veselo daļu: $\left\lfloor \frac{x}{10} \right\rfloor$ jeb nomet tā decimālpieraksta pēdējo ciparu.

1. Ja visas šīs kvantizētās vērtības būtu jāsūta, izmantojot vienādu, fiksētu bitu skaitu – cik lielu saspiešanas attiecību (*compression ratio* – sākotnējā faila izmēra attiecību pret saspiestā faila izmēru) varētu sasniegt?

2. Pieņemot, ka visas krāsu intensitātes (no 0 līdz 255) ir ar vienādām varbūtībām, kāda ir jaunās, kvantizētās ziņojumu virknes entropija? Kāda būtu teorētiski labākā iespējamā saspiešanas attiecība (ja izmantotu aritmētisko kodējumu vai citu optimālu metodi, kas tuvojas entropijas noteiktajai saspiešanas robežai).

1. Pēc kvantizēšanas var rasties jebkurš no skaitļiem $0, 1, 2, \ldots, 25$ – pavisam 26 vērtības. Katras vērtības nokodēšanai pietiek ar 5 bitiem, jo var izveidot $2^5 = 32$ dažādus 5-bitu kodus. Katrai no 26 vērtībām būs savs piecbitu kods (un vēl 6 piecbitu kodi paliks neizmantoti). Esam ieguvuši, ka ikvienu 8-bitu vērtību (no 0 līdz 255) pēc kvantizācijas var iekodēt 5 bitos. Šī ir zudumradošā saspiešana un failu saspiešanas attiecība (*compression ratio*) sanāk $8/5 = 1.6$.
   Piezīme. Praksē saspiešanas attiecība var sanākt nedaudz mazāka par 1.6, jo neviens fails nevar saturēt bitu skaitu, kas nedalās ar 8 (t.i. neveselu skaitu baitu). Tāpēc faila beigās daži biti ies zudumā (un, lai zinātu kur apstāties, var nākties ieviest kodējuma beigu simbolu [`EOT`] kā 27-to ziņojumu attēla beigās.)

2. Jebkurai kvantizācijas vērtībai no 0 līdz 24 ir vienāda varbūtība: $\frac{10}{256}$, bet pašai pēdējai vērtībai (25) ir varbūtība $\frac{6}{256}$. Entropiju iegūstam, saskaitot negatīvos logaritmus šīm varbūtībām, kas piereizināti ar pašām varbūtībām:

$$\underbrace{-\frac{10}{256}\log_2\left(\frac{10}{256}\right) - \ldots - \frac{10}{256}\log_2\left(\frac{10}{256}\right)}_{25\text{ saskaitāmie}} - \frac{6}{256}\log_2\left(\frac{6}{256}\right) =$$

$$= -25 \cdot \frac{10}{256}\log_2\left(\frac{10}{256}\right) - 1 \cdot \frac{6}{256}\log_2\left(\frac{6}{256}\right) \approx 4.695345.$$

Esam ieguvuši, ka taupīgāk kodējot šīs kvantificētās vērtības, tās var nosūtīt, tērējot vidēji 4.695345 bitus vienai vērtībai, nevis tieši 5 bitus, kā sanāca iepriekšējā punktā. Starp citu $\log_2(26) = 4.70044$, t.i. pat pieņemot, ka visām 26 kvantificētajām vērtībām ir vienādas varbūtības (nevis pašai pēdējai drusku mazāka varbūtība kā citām), kodējuma garums izmainās ļoti nedaudz.

$\square$

6.uzdevums (4 punkti): Spēlētājs X vēlas nosūtīt spēlētājam Y dažus ceturtās pakāpes polinomus ar veseliem koeficientiem:

$$f(x) = a_0x^4 + a_1x^3 + a_2x^2 + a_3x + a_4, \quad \text{kur} \quad a_0, \ldots, a_4 \in \mathbb{Z}.$$

Polinoma koeficientu a_i vietā spēlētājs X sūta s vērtības dažādiem veseliem argumentiem:

$$f(0), f(1), \ldots, f(s-1).$$

Vidū starp spēlētājiem X un Y atrodas ļaunprātīgais Šlopsterklopsters, kurš ne vairāk kā piecas no visām s nosūtītajām vērtībām drīkst nomainīt ar citiem skaitļiem (bet drīkst nomainīt arī mazāku skaitu vērtību vai nenomainīt nevienu).
Uzrakstīt nevienādību attiecībā pret parametru s (un atrisināt to), lai uzzinātu mazāko polinoma vērtību skaitu s, kuram spēlētājs Y noteikti varēs atjaunot X'a sūtīto polinomu, lai kā arī nerīkotos Šlopsterklopsters.

Ievērosim, ka diviem **dažādiem** ceturtās pakāpes polinomiem $f(x)$ un $g(x)$ var sakrist vērtības ne vairāk kā četros punktos. (Šo polinomu starpība $f(x) - g(x)$ pati ir ne vairāk kā ceturtās pakāpes polinoms. Un tāpēc šai starpībai $f(x) - g(x) = 0$ var būt ne vairāk kā 4 dažādas saknes saskaņā ar algebras pamatteorēmu; sk. `https://en.wikipedia.org/wiki/Fundamental_theorem_of_algebra`.
Ja polinoma vērtību pārsūtīšanā Šlopsterklopsters drīkst ieviest ne vairāk kā $c = 5$ kļūdas, tad katriem diviem polinomiem $f(x)$ un $g(x)$, kurus X sūta spēlētājam Y, jānodrošina, lai vismaz 11 no pārsūtāmajām vērtībām (pirms to sabojāšanas) atšķirtos. Pretējā gadījumā, ja izrādītos, ka atšķiras tikai 10 vērtības, Šlopsterklopsters tieši pusi no pārsūtāmā polinoma $f(x)$ vērtībām aizstās ar $g(x)$ vērtībām (vai otrādi). Tad saņēmējs Y šos polinomus nevarēs atšķirt.
Tā kā dažādiem polinomiem $f(x)$ un $g(x)$ vērtības pie četrām argumentu vērtībām drīkst sakrist arī bez jebkādas datu sabojāšanas (algebras pamatteorēma), bet vismaz 11 vērtībām jābūt atšķirīgām, tad kopīgais pārsūtāmo vērtību skaits s apmierina nevienādību:

$$s - 4 \geq 11, \quad \text{jeb} \quad s \geq 15.$$

Lekciju konspektos šo nevienādību parasti pierakstījām šādi:

$$s - (k-1) \geq 2c + 1,$$

kur s – pārsūtāmo vērtību skaits; k – polinoma koeficientu skaits (un $k-1$ – polinoma pakāpe), bet c – maksimāli atļautais kļūdu skaits. Sk. `https://bit.ly/2MXyJxZ`.
Ja pārsūta mazāk kā 15 polinoma vērtības, tad var viegli izveidot pretpiemērus. Iedomāsimies, ka spēlētājs X pārsūta vērtības tikai 14 punktos: argumentiem $x \in \{0, 1, \ldots, 12, 13\}$. Definējam divus 4.pakāpes polinomus, katram no kuriem ir pieci koeficienti:

$$\left\{ \begin{array}{l} f(x) = (x-1)(x-5)(x-9)(x-13) = 1x^4 - 28x^3 + 254x^2 - 812x + 585, \\ g(x) = 2(x-1)(x-5)(x-9)(x-13) = 2x^4 - 56x^3 + 508x^2 - 1624x + 1170. \end{array} \right.$$

Tās konstruētas tā, lai četras to vērtības sakristu, ja $x = 1, 5, 9, 13$. Šlopsterklopsters pārsūta visas šīs četras sakrītošās vērtības, kā arī piecas $f(x)$ vērtības (zaļais grafiks Attēlā 3) un vēl piecas $g(x)$ vērtības (zilais grafiks Attēlā 3). Tā kā spēlētājs Y nezina, kuras vērtības tiks sabojātas, viņš no saņemtajiem sarkanajiem punktiem nevar viennozīmīgi secināt, vai spēlētājs X sūtīja polinomu $f(x)$ vai polinomu $g(x)$.
Tas nozīmē, ka $f(x)$ un $g(x)$ attiecīgo koeficientu pārraidīšanai ar Rīda-Solomona kļūdu korekcijas algoritmu, kur atļautas piecas kļūdas, jānosūta vismaz $s = 15$ vērtības.

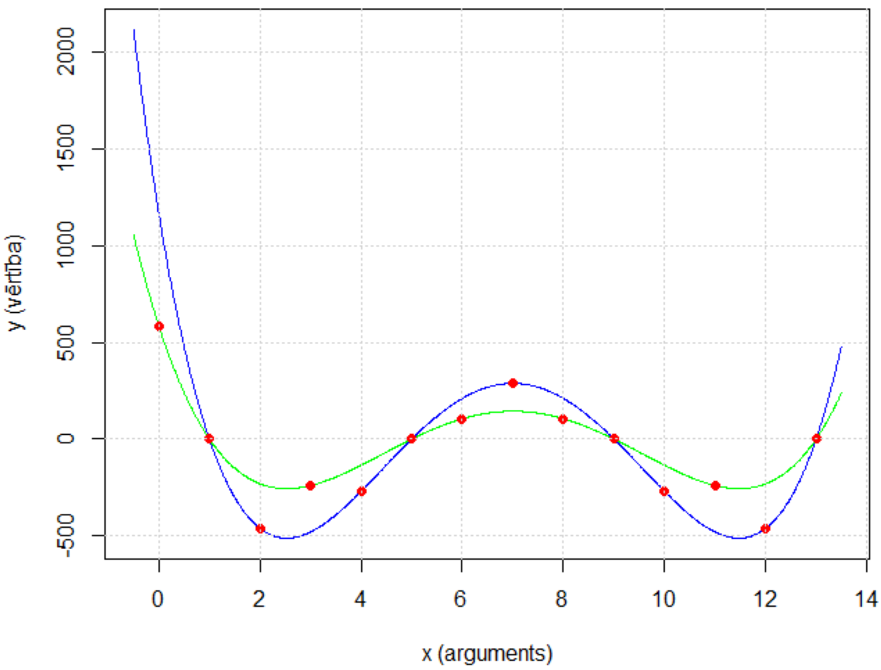


Figure 3: Sarkanie punkti neļauj atšķirt $f(x)$ un $g(x)$

□

7.uzdevums (2+2+2 punkti): (Ieteikums: Šajā uzdevumā var izmantot algebriskas identitātes par skaitļu kāpināšanu, Eilera teorēmu u.c. skaitļu teorijas rezultātus.)

1. Kriptogrāfijas algoritmam ir jāaprēķina $4^{143} \pmod{199}$ (atlikums, kas rodas dalot 4^{139} ar 199). Algoritms var ielūkoties reizināšanas tabulā pēc moduļa 199: ievadīt divus skaitļus a, b un saņemt to reizinājumu $a \cdot b \pmod{199}$ (ab atlikumu, dalot ar 199). Kādu mazāko reižu skaitu pietiek ielūkoties reizināšanas tabulā, lai atrastu $4^{139} \pmod{199}$.
   Piezīme. Algoritms var veikt arī dalīšanu ar atlikumu un atņemšanu, bet jāsaskaita tikai reizināšanas darbības. (Acīmredzot, pietiek veikt 142 reizināšanas; Jūsu uzdevums ir reizināšanu skaitu pēc iespējas samazināt.)

2. Kriptogrāfijas algoritmam ir jāaprēķina $4^{143} \pmod 7$. Algoritms var ielūkoties reizināšanas tabulā pēc moduļa 7. Kādu mazāko reizināšanu skaitu vajag, lai atrastu rezultātu?

3. Atrast $4^{143} \pmod 7$ vērtību; parādīt, kā tā iegūta.

1. Izsakām 143 binārajā pierakstā: $143_{10} = 128 + 8 + 4 + 2 + 1 = 10001111_2$. Lai uzzinātu, cik ir 4^{143}, sareizināsim $4^{128}4^8 4^4 4^2 4^1$ (tās ir 4 reizināšanas).
   Lai noskaidrotu skaitļus $4^2, 4^4, 4^8, 4^{16}, 4^{32}, 4^{64}, 4^{128}$, nepieciešamas vēl 7 reizināšanas (katru skaitli šajā virknē noskaidro, kāpinot iepriekšējo skaitli kvadrātā). Tātad, lai kāpinātu skaitļus 143 pakāpē, nepieciešamas pavisam $4 + 7 = 11$ reizināšanas darbības.

2. Lai uzzinātu $4^{143} \pmod 7$, vispirms pielietosim Eilera teorēmu. Šajā gadījumā kāpinātājs 143 ir daudz lielāks par moduli, kas ir 7, tāpēc augstām pakāpēm vērtības pēc moduļa 7 būs iecikl ējušās (un cikls ir $\varphi(7) = 6$).
   Tāpēc šoreiz vislabāk ir vispirms izdalīt 143 ar 6 (iegūstot atlikumu 5). Lai kāpinātu skaitli 5.pakāpē, nepieciešamas četras reizināšanas darbības.

3. Pēc Eilera teorēmas, katram skaitlim a, kas nedalās ar 7, ir spēkā $a^6 \equiv 1 \pmod 7$. Tai skaitā $4^6 \equiv 1 \pmod 7$:

$$4^{138+5} = 4^{6 \cdot 23+5} = (4^6)^{23} \cdot 4^5 \equiv 1^{23} \cdot 4^5 = 4^3 \cdot 4^2 = 64 \cdot 16.$$

   64 atlikums, dalot ar 7 ir 1, bet 16 atlikums, dalot ar 7 ir 2. To reizinājums ir $1 \cdot 2 = 2$.

□