V dalis
Kodavimas

Šios dalies pirmi du skyriai didele dalimi yra paruošti pagal V.Stakėno knygą „Informacijos kodavimas“ (VU leidykla, 1996).

Kodavimas plačiąja prasme — tai informacijos pakeitimas kodu (simbolių seka) arba vieno kodo pakeitimas kitu. Kodavimo tikslas — pritaikyti informacijos formą konkrečiam taikymui. Kodavimas nuo seno vaidina svarbų vaidmenį matematikoje.

Pavyzdžiai. 1. *Dešimtainė skaičiavimo sistema* — tai natūraliųjų skaičių kodavimo būdas. Romėniški skaičiai — kitas natūraliųjų skaičių kodavimo būdas.

2. *Dekartinės koordinatės* — tai būdas geometrines figūras užkoduoti skaičiais.

Su kompiuterių atsiradimu kodavimo reikšmė labai išaugo. Dabar tai pagrindinis uždavinys daugelyje informacinių technologijų sričių, pavyzdžiui,

- duomenų (skaičių, teksto, grafinių objektų) vaizdavimas kompiuterio atmintyje,
- informacijos apsauga,
- klaidų ištaisymas siunčiant duomenis nepatikimais ryšių kanalais,
- duomenų spaudimas duomenų bazėse.

Net patį programos rašymą kartais (ir, beje, visiškai teisingai) vadina kodavimu, o programos tekstą — kodu.

Šioje kurso dalyje mes susipažinsime su keliais svarbiausiais kodavimo teorijos uždaviniais: duomenų spaudimu, klaidas taisančiais kodais, kriptografija. Tai labai plačios diskrečiosios matematikos sritys, ir šiame kurse mes jas tik vos paliesime. Prieš tai dar supažindinsiu su kodavimo teorijos pagrindu — abėcėliniu kodavimu.

1 Abėcėlinis kodavimas

Šiame skyriuje šnekėsime apie tokius kodus, kur pranešimas skaidomas po vieną simbolį, ir kiekvienas simbolis užkoduojamas atskirai, nepriklausomai nuo kitų. Tokie kodai vadinami *abėcėliniais kodais*.

Abėcėle vadinsime pasirinktą baigtinę netuščią aibę $\mathcal{A}$, jos elementus vadinsime *simboliais (raidėmis)*. Baigtinės aibės $\mathcal{B}$ elementų skaičių žymėsime $|\mathcal{B}|$. Abėcėlę $\mathcal{A} = \{0, 1\}$ vadiname *dvinare* abėcėle.

1.1 apibrėžimas. *Baigtinę abėcėlės $\mathcal{A}$ simbolių seką $a_1a_2\cdots a_s$ vadinsime s* ilgio žodžiu. *Jei $\mathbf{x}$ yra žodis, tai $|\mathbf{x}|$ žymi jo ilgį. Jei $\mathbf{x}$, $\mathbf{y}$ yra du tos pačios abėcėlės žodžiai, tai $\mathbf{xy}$ žymi žodį, kuris gaunamas tiesiog sujungiant $\mathbf{x}$ ir $\mathbf{y}$. Abėcėlės $\mathcal{A}$ ilgio m žodžių aibę žymėsime $\mathcal{A}^m$, visų žodžių aibę — $\mathcal{A}^*$.*

1.2 pavyzdys. Tarkime, turime abėcėlę $\mathcal{A} = \{A, B, C, D\}$. Tada $\mathbf{x} = ABBA$ yra ilgio 4 žodis, $|\mathbf{x}| = 4$. Jei $\mathbf{y}$ pažymėsime žodį $ACDC$, tai $\mathbf{yx}$ bus žodis $ACDCABBA$. □

Lengva pastebėti, kad

$$\mathcal{A}^* = \bigcup_{m \geqslant 0} \mathcal{A}^m.$$

1.3 apibrėžimas. *Tegu $\mathcal{A}$, $\mathcal{B}$ yra dvi abėcėlės.* Abėcėliniu kodu *(arba tiesiog* kodu*) vadinsime injektyvų atvaizdį (injekciją) $c : \mathcal{A} \to \mathcal{B}^*$. Jei c yra kodas, tai jį pratęsime iki atvaizdžio $c^* : \mathcal{A}^* \to \mathcal{B}^*$ tokiu būdu:*

$$c^*(a_1 a_2 \cdots a_s) = c(a_1)c(a_2)\cdots c(a_s).$$

Abėcėlės $\mathcal{B}$ žodį $c^*(a_1 a_2 \cdots a_s)$ vadinsime žodžio $a_1 a_2 \cdots a_s$ kodu. *Abėcėlės $\mathcal{A}$ simbolių kodus $c(a)$, $a \in \mathcal{A}$, vadinsime* elementariais kodais.

1.4 pavyzdys. Tegu $\mathcal{A} = \{A, B, C\}$, $\mathcal{B} = \{0, 1\}$. Tarkime, kodas c yra toks: $c(A) = 0$, $c(B) = 1$ ir $c(C) = 01$. Tada $c^*(BCAA) = 10100$, t. y. žodžio $BCAA$ kodas yra žodis 10100. Žodžiai 0, 1 ir 01 yra elementarūs kodai. Matome, kad $c^*(AB) = c^*(C) = 01$, t. y. nors atvaizdis c yra injektyvus, atvaizdis c^* gali ir nebūti injektyvus. □

Dabar pateiksime įvairių kodų, kurie buvo arba yra naudojami, pavyzdžių.

1.5 pavyzdžiai. 1. *Graikų ugnies kodas.* Klasikinėje Graikijoje naudotas karinėms žinioms perduoti. Abėcėlę $\mathcal{A}$ sudaro 24 graikiškos raidės, $\mathcal{B} = \{1, 2, 3, 4, 5\}$, čia 1 žymi vieną degantį fakelą ir t. t. Raidė α užkoduojama 11, β — 12, ir t. t. Žodis mn buvo perduodamas uždegant po m ir n fakelų dviejose kalno viršūnės vietose.

2. *Cezario kodas.* Šį kodą pirmame amžiuje prieš Kristų naudojo karvedys Julijus Cezaris slaptiems pranešimams šifruoti. Abi abėcėlės $\mathcal{A}$ ir $\mathcal{B}$ sudarytos iš tų pačių lotyniškų raidžių. Raidė a keičiama raide $c(a)$, kuri abėcėlėje stovi trimis pozicijomis toliau. Pavyzdžiui, raidė A keičiama raide D, raidė B — raide E, ir t. t.

3. *Morzės kodas.* Jį sudarė amerikietis S. Morzė kaip priedą prie kito savo išradimo — telegrafo. Dviejų simbolių — taško ir brūkšnio — kombinacijomis koduojamos raidės ir skaičiai. Jiems atskirti dar naudojama pauzė. Kad tekstą būtų galima užkoduoti kuo trumpesne taškų ir brūkšnių seka, Morzė savo kodą sudarė taip, kad dažniau vartojamas raides atitiktų trumpesni kodai, o rečiau vartojamas — ilgesni. Pavyzdžiui, raidę e atitinka kodas tik iš vieno simbolio — taško, o raidę y — iš keturių: „−.−−“ .

4. *ASCII kodas.* Šis kodas (American Standard Code for Information Interchange) koduoja 128 simbolius (didžiąsias ir mažąsias raides, skaičius, skyrybos ženklus ir specialius simbolius) dvinarės abėcėles ilgio 7 žodžiais. ASCII kodo išplėstame variante ilgio 8 žodžiais koduojami 256 simboliai. Pavyzdžiui, raidės A ASCII kodas yra 1000001, raidės B — 1000010.

5. *Dešimtainių skaitmenų kodavimas.* Skaičius, užrašytus dešimtainėje sistemoje, dažnai tenka koduoti dvinarės abėcėlės žodžiais. Pateiksime keletą tokių kodų. Paprasčiausia tiesiog užrašyti skaitmenį dvejetainėje sistemoje, žr. 5 lentelės T stulpelį.

   Grėjaus (R.M. Gray) kodas pasižymi tuo, kad paeiliui einančių dešimtainių skaitmenų kodai skiriasi tik vienu simboliu (žr. G stulpelį).

skaitmuo	T	G	2 iš 5	ASCII
0	0000	0000	11000	0110000
1	0001	0001	00011	0110001
2	0010	0011	00101	0110010
3	0011	0010	00110	0110011
4	0100	0110	01001	0110100
5	0101	0111	01010	0110101
6	0110	0101	01100	0110110
7	0111	0100	10001	0110111
8	1000	1100	10010	0111000
9	1001	1101	10100	0111001

1 lentelė: Dešimtainių skaitmenų kodavimas

Kodas „2 iš 5“ priskiria skaitmenims penkių simbolių dvinarius žodžius; lygiai du simboliai kiekviename kodo žodyje yra vienetukai (žr. „2 iš 5“ stulpelį). Šis kodas gali būti naudojamas klaidoms aptikti.

Paskutiniame stulpelyje surašyti dešimtainių skaitmenų ASCII kodai. □

1.6 apibrėžimas. *Žodį* $\mathbf{x} \in \mathcal{A}^*$ *vadinsime žodžio* $\mathbf{y} \in \mathcal{A}^*$ priešdėliu *(arba* prefiksu*), jei egzistuoja toks žodis* $\mathbf{z} \in \mathcal{A}^*$*, kad* $\mathbf{y} = \mathbf{xz}$.

Jei $\mathbf{x}$ *yra žodžio* $\mathbf{y}$ *priešdėlis, tai žymėsime* $\mathbf{x} \prec \mathbf{y}$. *Jei dviejų žodžių* $\mathbf{x}$ *ir* $\mathbf{y}$ *m ilgio priešdėlis yra tas pats, tai rašysime* $\mathbf{x} \sim_m \mathbf{y}$.

Tuščią žodį $\varnothing$, neturintį nei vieno simbolio, natūralu laikyti kiekvieno žodžio priešdėliu.

1.7 pavyzdys. Žodžio „diskas“ priešdėliai yra tuščias žodis $\varnothing$, žodžiai „d“, „di“, „dis“, „disk“, „diska“ ir „diskas“. Be to, „diskas“ $\sim_4$ „diskretus“ (žinoma, taip pat ir „diskas“ $\sim_3$ „diskretus“ ir t. t.). □

1.8 užduotis. Įrodykite, kad sąryšiai „$\prec$“ ir „$\sim_m$“ yra atitinkamai negriežtos tvarkos ir ekvivalentumo sąryšiai aibėje $\mathcal{A}^*$. (Įrodymas labai paprastas, užtenka patikrinti negriežtos tvarkos ir ekvivalentumo sąryšių apibrėžimus.)

Jei $c : \mathcal{A} \to \mathcal{B}^*$ yra kodas, tai c yra injektyvus atvaizdis. Tačiau iš to neišplaukia, kad jo tęsinys $c^* : \mathcal{A}^* \to \mathcal{B}^*$ taip pat injektyvus.

1.9 apibrėžimas. *Kodą* $c : \mathcal{A} \to \mathcal{B}^*$ *vadiname* iššifruojamu*, jei jo tęsinys* $c^* : \mathcal{A}^* \to \mathcal{B}^*$ *yra injektyvus atvaizdis.*

1.10 pavyzdys. Tegu $\mathcal{A} = \{A, B, C\}$, $\mathcal{B} = \{0, 1\}$. Tada kodas

$$c(A) = 00,\ c(B) = 10,\ c(C) = 11 \tag{1}$$

yra iššifruojamas[1], taip pat iššifruojamas ir kodas

$$c(A) = 0,\ c(B) = 01,\ c(C) = 0011, \tag{2}$$

o kodas

$$c(A) = 0,\ c(B) = 01,\ c(C) = 010$$

nėra toks, nes, pavyzdžiui, $c^*(BA) = c^*(C) = 010$. □

[1]Nėra paprasta nustatyti, ar kodas iššifruojamas, ar ne. Šiame kurse šio klausimo plačiau neliesime.

Pastebėkime svarbų iššifruojamų kodų (1) ir (2) skirtumą. Jeigu mums perduodamas (1) kodu užkoduotas abėcėlės $\mathcal{A}$ žodis ir mes gaunamą seką skaitome simbolis po simbolio, tai siunčiamą užkoduotą simbolį galėsime atpažinti tuoj pat, kai tik perskaitysime paskutinį jo elementaraus kodo ženklą. Tačiau taip nebus, jei naudojamas (2) kodas. Pavyzdžiui, tik perskaitę visą siunčiamą seką 001, galime nuspręsti, kad pirmąjį simbolį reikia iššifruoti „A“. Kodą, kurio kiekvieną elementarų kodą galima atpažinti (dekoduoti) vos jį perskaičius, vadinsime *p-kodu*. Formalus jo apibrėžimas yra kiek kitoks, bet reiškia tą patį:

1.11 apibrėžimas. *Kodą $c : \mathcal{A} \to \mathcal{B}^*$ vadinsime* p-kodu, *jei joks elementarus kodas $c(a)$, $a \in \mathcal{A}$, nėra kito elementaraus kodo $c(a')$, $a' \in \mathcal{A}$, $a' \neq a$, priešdėlis.*

1.12 pavyzdys. (2) kodas yra iššifruojamas, tačiau nėra p-kodas. □

1.13 teorema. *Bet kuris p-kodas yra iššifruojamas.*

Įrodymas. Tarkime priešingai: kodas $c : \mathcal{A} \to \mathcal{B}^*$ yra p-kodas, bet ne iššifruojamas. Taigi kodo c tęsinys $c^* : \mathcal{A}^* \to \mathcal{B}^*$ nėra injekcija, t. y., egzistuoja du skirtingi $\mathcal{A}^*$ žodžiai $a_{i_1}a_{i_2}\cdots a_{i_k}$ ir $a_{j_1}a_{j_2}\cdots a_{j_l}$, kuriuos c^* atvaizduoja į tą patį žodį b, t. y.

$$c^*(a_{i_1}a_{i_2}\cdots a_{i_k}) = b_{i_1}b_{i_2}\cdots b_{i_k} = b = b_{j_1}b_{j_2}\cdots b_{j_l} = c^*(a_{j_1}a_{j_2}\cdots a_{j_l}),$$

čia b_i yra simbolio $a_i \in \mathcal{A}$ elementarus kodas, t. y. $b_i = c(a_i) \in \mathcal{B}^*$ $\forall i$. Tegu t yra mažiausias toks indeksas, kad $b_{i_t} \neq b_{j_t}$, t. y. $b_{i_1} = b_{j_1}, b_{i_2} = b_{j_2}, \ldots, b_{i_{t-1}} = b_{j_{t-1}}$, o $b_{i_t} \neq b_{j_t}$. Tada $b_{i_t}b_{i_{t+1}}\cdots b_{i_k} = b_{j_t}b_{j_{t+1}}\cdots b_{j_l}$, todėl egzistuoja toks žodis $b' \in \mathcal{B}^*$, kad $b_{i_t} = b_{j_t}b'$ (jei b_{i_t} yra ilgesnis už b_{j_t}) arba $b_{i_t}b' = b_{j_t}$ (jei b_{i_t} yra trumpesnis už b_{j_t}), o tai prieštarauja tam, kad kodas c yra p-kodas. □

1.14 teorema. (Krafto-Makmilano) *Tegu abėcėlės $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$, $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$, ir tegu duoti m teigiamų sveikųjų skaičių $d_1, d_2, \ldots, d_m$. Toks iššifruojamas kodas $c : \mathcal{A} \to \mathcal{B}^*$, kad $|c(a_i)| = d_i$ kiekvienam i, egzistuoja tada ir tik tada, kai*

$$\sum_{i=1}^{m} \frac{1}{n^{d_i}} \leqslant 1. \tag{3}$$

Be to, bent vienas iš tokių kodų yra p-kodas.

Be įrodymo.

Kaip sudaryti tokį p-kodą iš teoremos? Pailiustruosime sudarymą pavyzdžiu.

1.15 pavyzdys. Tegu $\mathcal{A} = \{A, B, C, D, E, F, G\}$, $\mathcal{B} = \{0, 1, 2\}$, ir norime sudaryti p-kodą $c : \mathcal{A} \to \mathcal{B}^*$ tokį, kad $|c(A)| = 1$, $|c(B)| = 1$, $|c(C)| = 2$, $|c(D)| = 2$, $|c(E)| = 3$, $|c(F)| = 3$ ir $|c(G)| = 3$. Nesunku patikrinti, kad šie skaičiai tenkina (3) sąlygą:

$$\sum_{i=1}^{7} \frac{1}{3^{d_i}} = \frac{1}{3} + \frac{1}{3} + \frac{1}{3^2} + \frac{1}{3^2} + \frac{1}{3^3} + \frac{1}{3^3} + \frac{1}{3^3} = 1 \leqslant 1,$$

todėl toks p-kodas egzistuoja. Jį konstruojame tokiu būdu:

- Iš pradžių parenkame trumpiausio ilgio elementarius kodus, šiuo atveju ilgio 1 elementarius kodus $c(A)$ ir $c(B)$. Paprastumo dėlei, priskirkime jiems pirmus abėcėlės $\mathcal{B}$ elementus: $c(A) = 0$, $c(B) = 1$.

- Parinksime ilgio 2 elementarius kodus, šiuo atveju $c(C)$ ir $c(D)$. Jų priešdėliai negali būti jau panaudoti elementarūs kodai, t. y. 0 ir 1, kitaip negausime p-kodo. Taigi pirmas simbolis turi būti iš aibės $\mathcal{B} \setminus \{0,1\}$, šiuo atveju tai gali būti tik 2. Antru simboliu galime imti bet kurį abėcėlės $\mathcal{B}$ elementą. Vėlgi imkime pirmus abėcėlės $\mathcal{B}$ elementus, nors tai ir nėra būtina: $c(C) = 20$ ir $c(D) = 21$.

- Dabar ilgio 3 elementarūs kodai. Priešdėlių 0, 1, 20 ir 21 nebegalime naudoti, lieka tik priešdėlis 22. Trečiu simboliu vėlgi galime parinkti bet kurį abėcėlės $\mathcal{B}$ elementą. Gauname $c(E) = 220$, $c(F) = 221$, $c(G) = 222$.

- Gavome reikiamą kodą. Atkreipkite dėmesį, kad paskutiniame etape panaudojome visus abėcėlės $\mathcal{B}$ elementus. Bendru atveju, jei (3) sąlygoje nelygybė yra griežta, gali likti nepanaudotų simbolių. □

1.16 pastaba. Krafto-Makmilano teorema kai kuriais atvejais (bet ne visada!) leidžia parodyti, kad duotas kodas nėra iššifruojamas. Jei (3) nelygybė to kodo elementarių kodų ilgiams nėra patenkinta, tai iššifruojamas kodas su tokiais ilgiais egzistuoti negali, todėl duotas kodas nėra iššifruojamas. Bet ne atvirkščiai: jei (3) nelygybė duoto kodo elementarių kodų ilgiams yra patenkinta, tai dar nereiškia, kad duotas kodas yra iššifruojamas. Tai tik reiškia, kad *egzistuoja* iššifruojamas kodas su tokiais elementarių kodų ilgiais, bet nieko nesako apie duoto kodo iššifruojamumą.

1.17 pavyzdys. Ar iššifruojamas kodas $c : \mathcal{A} \to \mathcal{B}^*$, kur $\mathcal{A} = \{A, B, C, D, E\}$, $\mathcal{B} = \{0, 1\}$, $c(A) = 0$, $c(B) = 01$, $c(C) = 100$, $c(D) = 101$, $C(E) = 110$? Matome, kad (3) nelygybė nėra patenkinta, nes

$$\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} = \frac{9}{8} > 1,$$

todėl šis kodas negali būti iššifruojamas. Iš tikro, lengva pastebėt, kad $c^*(AD) = c^*(BB) = 0101$, todėl iššifruojamo kodo apibrėžimas netenkinamas. □

2 Optimalus abėcėlinis kodavimas

Praktikoje norima, kad abėcėliniu kodu užkoduotas pranešimas būtų kuo trumpesnis.

2.1 Optimalus kodas

Tarkime, turime abėcėlę $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ ir konkretų iššifruojamą kodą $c : \mathcal{A} \to \mathcal{B}^*$. Pažymėkime $b_i = c(a_i) \in \mathcal{B}^*$, $1 \leqslant i \leqslant m$. Tada bet kuris kodas, gautas sukeitus vietomis kodo c elementarius kodus $b_1, b_2, \ldots, b_m$, irgi bus iššifruojamas.

2.1 pavyzdys. Tarkime, turime abėcėles $\mathcal{A} = \{A, B, C\}$, $\mathcal{B} = \{0, 1\}$, ir iššifruojamą kodą c, apibrėžtą tokiais elementariais kodais: $c(A) = 0$, $c(B) = 10$, $c(C) = 11$. Tada sukeitę elementarius kodus vietomis, irgi gausime iššifruojamą kodą c', pavyzdžiui, $c'(A) = 10$, $c'(B) = 11$, $c'(C) = 0$. □

Jei elementarių kodų ilgiai vienodi, tai sukeitus juos vietomis, užkoduoto pranešimo ilgis nepasikeis. Bet jei jie skirtingi, tai užkoduoto pranešimo ilgis priklauso nuo to, kiek kokių simbolių yra pranešime S, kurį reikia užkoduoti, ir kokie elementarūs kodai kokiems abėcėles $\mathcal{A}$ simboliams yra priskirti. Jei turime konkretų pranešimą ir konkretų kodą, tai nesunku taip sukeisti vietomis elementarius kodus, kad užkoduoto pranešimo ilgis būtų trumpiausias.

Tarkime, simbolis a_1 pranešime S pasirodo k_1 kartų, simbolis a_2 — k_2 kartus, ir t. t. Kiekvienam i elementaraus kodo b_i ilgį pažymėkime l_i. Tarkime, $k_j \geqslant k_i$ (a_j pranešime S pasirodo dažniau, negu a_i) ir $l_j \leqslant l_i$ (elementaraus kodo b_j ilgis mažesnis, nei b_i). Tada, užkodavę simbolį a_j elementariu kodu b_j, o simbolį a_i elementariu kodu b_i, gauname kodą, kurio ilgis $k_j l_j + k_i l_i$, tuo tarpu užkodavę atvirkščiai gauname kodą, kurio ilgis $k_j l_i + k_i l_j$. Nesunku

pastebėti, kad $k_i l_i + k_j l_j \leqslant k_i l_j + k_j l_i$, todėl kodas trumpesnis, jei dažniau pasitaikantį simbolį a_j koduojame trumpesniu elementariu kodu.

Iš tikrųjų, tegu $k_j = k_i + a$, $l_i = l_j + b$, $a, b \geqslant 0$. Tada

$$\begin{aligned}(k_i l_j + k_j l_i) - (k_i l_i + k_j l_j) &= (k_i l_j + (k_i + a)(l_j + b)) - (k_i (l_j + b) + (k_i + a) l_j) && (4)\\ &= (k_i l_j + k_i l_j + k_i b + a l_j + ab) - (k_i l_j + k_i b + k_i l_j + a l_j) && (5)\\ &= ab \geqslant 0. && (6)\end{aligned}$$

Todėl, kad gautume trumpiausią užkoduotą pranešimą, elementarius kodus abėcėlės $\mathcal{A}$ simboliams priskiriame tokiu būdu: išrikiuojame abėcėlės $\mathcal{A}$ simbolius taip, kad dažniau pranešime S pasirodantys simboliai stovėtų prieš rečiau pasirodančius simbolius, o elementarius kodus išrikiuojame jų ilgių didėjimo tvarka, ir priskiriame juos simboliams šia tvarka. Trumpiau tariant, dažniau pasitaikančius simbolius koduojame trumpesniais žodžiais, o rečiau — ilgesniais.

2.2 pavyzdys. Imkime kodą iš pereito pavyzdžio ir pranešimą $S = ACBBCBBC$. Dažniausia raidė yra B, po to eina C ir A. Elementarūs kodai, išrikiuoti ilgių didėjimo tvarka, bus $0, 10, 11$. Taigi elementarius kodus simboliams priskiriame taip: $c(B) = 0$, $c(C) = 10$, $c(A) = 11$. Užkodavę šiuo kodu žodį S, gausime žodį $c(S) = 111000100010$, kurio ilgis 12. Su šiais elementariais kodais mes negalėtume užkoduoti žodžio S trumpesniu kodu. □

2.3 pastaba. Šis paprastas metodas leidžia minimizuoti kodo ilgį tik turint fiksuotą pranešimą S ir fiksuotą kodą c.

Ką daryti, kai reikia parinkti trumpiausią kodą, dar nežinant pranešimo, kurį reiks užkoduoti, ir dėl to negalime suskaičiuoti, kiek kokių simbolių jame yra? Dažnai žinome bent kiekvieno simbolio pasirodymo būsimuose pranešimuose dažnius, arba, kitaip sakant, tikimybes.

2.4 apibrėžimas. Informacijos šaltiniu *vadinsime abėcėlę* $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ *su skaičių* $p_1, p_2, \ldots, p_m$ *rinkiniu, tenkinančiu savybes:*

$$\sum_{i=1}^{m} p_i = 1, \quad 0 \leqslant p_i \leqslant 1 \ \forall i.$$

Šaltinis mums pateikia abėcėlės $\mathcal{A}$ simbolių sekas, o skaičiai $p_1, p_2, \ldots, p_m$ reiškia abėcėlės simbolių $a_1, a_2, \ldots, a_m$ pasirodymo tikimybes. Laikome, kad šaltinis sugeneruoja eilinį simbolį su šiomis tikimybėmis visiškai nepriklausomai nuo prieš tai buvusių ir paskui eisiančių simbolių (toks šaltinis vadinamas *be atminties*). Toks šaltinio modelis gerai vaizduoja atsitiktinių simbolių sekos generavimą, bet, pavyzdžiui, nelabai tinka žmonių kalbai modeliuoti, ir visiškai netinka vaizdams modeliuoti.

2.5 pavyzdys. Šaltinis, kurio abėcėlė $\mathcal{A} = \{A, B, C\}$ su tikimybėmis $p_1 = 0{,}5$, $p_2 = 0{,}3$, $p_3 = 0{,}2$. Tai reiškia, kad šis šaltinis gali sugeneruoti tris simbolius: A, B, C su tokiomis tikimybėmis: tikimybė, kad pasirodys simbolis A, yra 0,5, kad simbolis B — 0,3, kad simbolis C — 0,2. □

Tarkime, turime informacijos šaltinį su abėcėle $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ ir tikimybėmis $p_1, p_2, \ldots, p_m$, bei iššifruojamą kodą $c : \mathcal{A} \to \mathcal{B}^*$. Apibrėžkime vidutinį šio kodo elementarių kodų ilgį $l(c)$ (duotam šaltiniui):

$$l(c) = \sum_{i=1}^{m} p_i \left| c(a_i) \right|.$$

Tai vidutinis abėcėlės $\mathcal{B}$ simbolių skaičius, kurio reikia vienam abėcėlės $\mathcal{A}$ simboliui užkoduoti.

2.6 pavyzdys. Imkime šaltinį iš pereito pavyzdžio: abėcėlę $\mathcal{A} = \{A, B, C\}$ su tikimybėmis $p_1 = 0{,}5$, $p_2 = 0{,}3$, $p_3 = 0{,}2$. Imkime abėcėlę $\mathcal{B} = \{0, 1\}$ ir iššifruojamą kodą c, apibrėžtą taip: $c(A) = 11$, $c(B) = 10$, $c(C) = 0$. Tada $l(c) = 0{,}5 \cdot 2 + 0{,}3 \cdot 2 + 0{,}2 \cdot 1 = 1{,}8$. Tai reiškia, kad jei imtume kažkokį šio šaltinio sugeneruotą 10 simbolių ilgio pranešimą, tai jį užkodavus šiuo kodu, vidutinis užkoduoto pranešimo ilgis būtų 18 simbolių. Dabar imkime iššifruojamą kodą c', apibrėžtą taip: $c'(A) = 0$, $c'(B) = 10$, $c'(C) = 110$. Tada $l(c') = 0{,}5 \cdot 1 + 0{,}3 \cdot 2 + 0{,}2 \cdot 3 = 1{,}7$. Taigi duotam šaltiniui antrasis kodas geresnis, nors jo elementarūs kodai ilgesni už pirmojo. □

Kadangi bet kokį iššifruojamą kodą galime pakeisti p-kodu, kuris turi tiek pat elementarių kodų ir jų ilgiai tie patys (žr. Krafto-Makmilano teoremą), tai toliau apsiribosime pastaraisiais.

2.7 apibrėžimas. *p-kodą $\bar{c} : \mathcal{A} \to \mathcal{B}^*$ vadinsime* optimaliu *(duotam šaltiniui), jei $l(\bar{c}) = \min_c l(c)$, čia minimumas imamas pagal visus p-kodus $c : \mathcal{A} \to \mathcal{B}^*$.*

2.8 teorema. *Kiekvienam šaltiniui egzistuoja optimalus kodas.*

Įrodymas. Tegu $p_* = \min p_i$, $s_i = |c(a_i)|$ $\forall i$, o $N \geqslant 1$ — bet koks skaičius. Nesunku įsitikinti, kad bet kokiam kodui c, kuriam $\max s_i \geqslant N/p_*$, galioja nelygybė $l(c) \geqslant N$. Iš tikro,

$$l(c) = \sum_{i=1}^{m} p_i s_i \geqslant p_* \sum_{i=1}^{m} s_i \geqslant p_* \max s_i \geqslant p_* \cdot N/p_* = N.$$

Tegu dabar $\hat{c}$ yra koks nors p-kodas, $\hat{l} = l(\hat{c})$. Tada optimalaus kodo reikia ieškoti kodų, tenkinančių sąlygą $\max s_i \leqslant \hat{l}/p_*$, aibėje. Kadangi ši aibė baigtinė, tai optimalus kodas tikrai egzistuoja. □

2.2 Šenono–Fano kodas

Kaip sudaryti optimalų kodą? Iš pradžių parodysime, kaip gauti Šenono–Fano kodą (C.E. Shannon, R.M. Fano), kuris dažnai yra beveik optimalus, ir yra nesunkiai sudaromas (lengviau, nei optimalus kodas). Tegu $m = |\mathcal{A}|$, $n = |\mathcal{B}|$. Kodo $c : \mathcal{A} \to \mathcal{B}^*$ elementariems kodams parinkime tokius ilgius $s_1, s_2, \ldots, s_m$, kad būtų

$$\frac{1}{n^{s_i}} \leqslant p_i < \frac{1}{n^{s_i - 1}}, \quad i = 1, \ldots, m.$$

p-kodas, kurio elementarių kodų ilgiai tenkina šias nelygybes, vadinamas *(n-nariu) Šenono–Fano kodu*. Jis visada egzistuoja, nes tokiu būdu parinkti ilgiai tenkina Krafto–Makmilano nelygybę:

$$\sum_{i=1}^{m} \frac{1}{n^{s_i}} \leqslant \sum_{i=1}^{m} p_i = 1.$$

Turėdami žodžių ilgius, patį p-kodą sudarome pagal pereito skyriaus pabaigoje aprašytą algoritmą.

2.9 pavyzdys. Šaltinis tegu būna toks: $\mathcal{A} = \{A, B, C, D\}$ su tikimybėmis $p_1 = 0{,}4$, $p_2 = 0{,}3$, $p_3 = 0{,}2$, $p_4 = 0{,}1$. Abėcėlė $\mathcal{B} = \{0, 1\}$. Tada Šenono–Fano kodo žodžių ilgiai bus tokie: s_1 bus 2, nes $2^{-2} \leqslant 0{,}4 < 2^{-1}$, s_2 irgi bus 2, s_3 bus 3, nes $2^{-3} \leqslant 0{,}2 < 2^{-2}$, ir s_4 bus 4. Ir kodas galėtų būti toks: $c(A) = 00$, $c(B) = 01$, $c(C) = 100$, $c(D) = 1010$. Jo $l(c) = 2{,}4$. Šis kodas nėra optimalus. Netrukus išmoksime sudaryti optimalų kodą ir įsitikinsime, kad optimalus būtų, pavyzdžiui, toks kodas c': $c'(A) = 1$, $c'(B) = 00$, $c'(C) = 010$, $c'(D) = 011$, kurio $l(c') = 1{,}9$ (žr. 2.10 pavyzdį). □

2.3 Hafmeno kodas

Dabar panagrinėsime algoritmą, kuris duotam šaltiniui visada sudaro optimalų kodą. Tai *Hafmeno* (D.A. Huffman) *algoritmas*. Juo remiantis sudaryti kodai vadinami *Hafmeno kodais* (*n-nariais* Hafmeno kodais, kai norėsime pabrėžti, kiek simbolių naudojama koduojant, čia $n = |B|$).

Iš pradžių panagrinėsime kodavimo dvinarės abėcėlės žodžiais atvejį.

Turime informacijos šaltinį S su abėcėle $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ ir tikimybėmis $p_1, p_2, \ldots, p_m$. Algoritmas Hafmeno kodui c sudaryti būtų toks. Pernumeruokime abėcėlės $\mathcal{A}$ elementus taip, kad $p_1 \geqslant p_2 \geqslant \cdots \geqslant p_m$.

- Jei $m = 1$, tai $c(a_1) = 0$.
- Jei $m = 2$, tai $c(a_1) = 0$, $c(a_2) = 1$.
- Jei $m > 2$, tai mažiname abėcėlę $\mathcal{A}$ vienu simboliu tokiu būdu: pažymėkime S' šaltinį su abėcėle $\mathcal{A}' = \{a_1, a_2, \ldots, a_{m-2}, b\}$ ir tikimybėmis $p_1, p_2, \ldots, p_{m-2}, p_{m-1} + p_m$, t. y. sujungiame du simbolius su mažiausiomis tikimybėmis į vieną, o jų tikimybes sudedame. Taip darome, kol gauname abėcėlę iš dviejų simbolių. Tada optimalus kodas yra akivaizdus: vieną simbolį reikia koduoti 0, kitą — 1. Grįžtame atgal, iš abėcėlės $\mathcal{A}'$ optimalaus kodo c' sudarydami abėcėlės $\mathcal{A}$ optimalų kodą c tokiu būdu:

$$\begin{aligned} c(a_i) &= c'(a_i) \quad \forall i, 1 \leqslant i \leqslant m-2, \\ c(a_{m-1}) &= c'(b)0, \\ c(a_m) &= c'(b)1, \end{aligned}$$

  t. y. iš simbolio b kodo gauname kodus simboliams a_{m-1} ir a_m, tiesiog prirašydami atitinkamai 0 ir 1, o kitų simbolių a_i kodai lieka tokie patys.

2.10 pavyzdys. Sudarykime Hafmeno kodą šaltiniui iš 2.9 pavyzdžio. Šaltinio abėcėlė $\mathcal{A} = \{A, B, C, D\}$, ir tikimybės $p_1 = 0{,}4$, $p_2 = 0{,}3$, $p_3 = 0{,}2$, $p_4 = 0{,}1$. Abėcėlė $\mathcal{B} = \{0, 1\}$. Hafmeno kodo sudarymą vaizduosime lentele. Kad būtų mažiau rašymo, pačių abėcėlės $\mathcal{A}$ simbolių nerašysime, o tik jų tikimybes. Stulpelyje $\mathcal{A}_k$ surašytos k-tajame žingsnyje gautos abėcėlės $\mathcal{A}_k$ simbolius atitinkančios tikimybės, žvaigždutės iš dešinės žymi, kurie simboliai kitu žingsniu bus jungiami į vieną, pabraukimas reiškia, kad tikimybė (simbolis) ankstesniu žingsniu yra gauta iš dviejų. Stulpeliai c_k bus naudojami grįžtant užrašyti abėcėlės $\mathcal{A}_k$ simbolių kodams. Taigi, vis sudėdami po dvi mažiausias tikimybes ir išrikiuodami gautas tikimybes mažėjimo tvarka, gauname tokią lentelę:

$\mathcal{A}_1$	c_1	$\mathcal{A}_2$	c_2	$\mathcal{A}_3$	c_3
0,4		0,4		$\underline{0{,}6}$	
0,3		$0{,}3^*$		$\overline{0{,}4}$	
$0{,}2^*$		$\underline{0{,}3^*}$			
$0{,}1^*$					

Dabar grįždami sukonstruosime Hafmeno kodus kiekvienai iš abėcėlių $\mathcal{A}_3$, $\mathcal{A}_2$ ir $\mathcal{A}_1$. Abėcėlėje $\mathcal{A}_3$ yra tik du simboliai, todėl vieną iš jų užkoduojame 0, kitą — 1. Abėcėlės $\mathcal{A}_2$ simbolių kodus gauname taip: iš to abėcėlės $\mathcal{A}_3$ simbolio, kuris buvo gautas sujungus du abėcėlės $\mathcal{A}_2$ simbolius, kodo gauname tų dviejų sujungtų simbolių kodus, prirašydami atitinkamai 0 ir 1. Analogiškai gauname abėcėlės $\mathcal{A}_1 = \mathcal{A}$ simbolių kodus, t. y. gauname ieškomą Hafmeno kodą. Lentelė pasidarys tokia:

$\mathcal{A}_1$	c_1	$\mathcal{A}_2$	c_2	$\mathcal{A}_3$	c_3
0,4	1	0,4	1	$\underline{0{,}6}$	0
0,3	00	0,3*	00	0,4	1
0,2*	010	$\underline{0{,}3^*}$	01		
0,1*	011				

Matome, kad gavome tą kodą, kuris pateiktas 2.9 pavyzdyje. □

Algoritmą nesunku pakeisti taip, kad jis tiktų Hafmeno kodui n-narės abėcėlės žodžiais sudaryti. Mažinant abėcėlę, vienu simboliu reikia keisti ne porą, bet n simbolių su mažiausiomis tikimybėmis. Tiesa, gali atsitikti, kad po paskutinio sumažinimo gausime abėcėlę iš mažiau nei n simbolių. Siekdami, kad jų liktų lygiai n (nes tik tokiu atveju kodas bus optimalus), nustatykime, kiek simbolių reikia sujungti pirmu žingsniu.

Tegu $|\mathcal{A}| = m$, $|\mathcal{B}| = n$, ir abėcėlės mažinimo procese atlikome s žingsnių, pirmajame sujungdami $2 \leqslant u \leqslant n$ simbolių, o visuose kituose po n. Tada $m = (u-1) + (s-1)(n-1) + n = s(n-1) + u$, arba

$$u \equiv m \bmod (n-1), \quad 2 \leqslant u \leqslant n.$$

Ši sąlyga vienareikšmiškai apibrėžia pirmu žingsniu sujungiamų simbolių skaičių u. Primenu, kad $a \equiv b \bmod c$ reiškia, kad $a - b$ dalijasi iš c be liekanos, arba, kitaip pasakius, kad a padaliję iš c gausime tą pačią liekaną, kaip ir b padaliję iš c.

2.11 pavyzdys. Sudarysime trinarį Hafmeno kodą šaltiniui su tikimybėmis 0,4; 0,2; 0,2; 0,1; 0,05; 0,05. Visų pirma apskaičiuokime, kiek pirmu žingniu reikės sujungti simbolių. Šiuo atveju $m = 6$, $n = 3$, todėl u apskaičiuosime iš sąlygų $u \equiv 6 \bmod 2$ ir $2 \leqslant u \leqslant 3$. Iš pirmos sąlygos gauname, kad u yra lyginis (nes u padaliję iš 2 turime gauti tą pačią liekaną, kaip ir 6 padaliję iš 2), todėl iš antros nusprendžiame, kad $u = 2$. Taigi pirmu žingsniu sujungsime 2 simbolius, o kitais žingsniais — po 3. Lentelė bus tokia:

$\mathcal{A}_1$	c_1	$\mathcal{A}_2$	c_2	$\mathcal{A}_3$	c_3
0,4	1	0,4	1	$\underline{0{,}4}$	0
0,2	2	0,2	2	0,4	1
0,2	00	0,2*	00	0,2	2
0,1	01	0,1*	01		
0,05*	020	$\underline{0{,}1^*}$	02		
0,05*	021				

□

2.12 teorema. *Hafmeno kodai yra optimalūs.*

Be įrodymo.

3 Duomenų spaudimas

3.1 Įvadas

Ankstesniuose skyriuose tyrinėjome abėcėlinį kodavimą, kai kiekvienas simbolis yra koduojamas atskirai ir naudojamasi jų pasirodymo tikimybėmis. Matėme, kad taip darant, neįmanoma užkoduoti trumpesniu kodu, negu tai daro optimalus Hafmeno kodas. Šiame skyriuje panagrinėsime,

kokius neabėcėlinius kodus galima naudoti duomenims spausti, kad suspaustume duomenis labiau, nei leidžia optimalus abėcėlinis kodavimas.

Tarkime, turime kažkokį pranešimą, kuriuo nors visuotinai priimtu būdu užkoduotą (pavyzdžiui, tekstas paprastai koduojamas ASCII kodu) ir saugomą kompiuterio atmintyje. Dažnai tas užkodavimas nėra optimalus. Pavyzdžiui, ASCII kodas kiekvienam iš 256 simbolių priskiria 8 bitų ilgio žodį (bitas — tai simbolis iš abėcėlės $\mathcal{B} = \{0, 1\}$). Bet paprastai tekstuose jų naudojama žymiai mažiau, negu 256 (priklausomai nuo kalbos — apie 60–80, įskaitant didžiąsias ir mažąsias raides, skaitmenis, skyrybos ženklus). Be to, simbolių pasirodymo tekste tikimybės yra skirtingos, ir kiekvienai kalbai yra žinoma, kaip dažnai pasikartoja duotas simbolis šia kalba parašytame tekste. Taigi galima pasirinkti mažesnę abėcėlę su jos simbolių pasirodymo dažniais ir sudaryti optimalų abėcėlinį kodą koduoti šia kalba parašytus tekstus. Žinant simbolių pasirodymų dažnius, nesunku apskaičiuoti, kad dažniausiai pasitaikančioms kalboms tokio kodo vidutinis elementarių kodų ilgis $l(c)$ būtų šiek tiek mažesnis už 6, t. y. palyginus su ASCII kodu užkoduoto teksto ilgis sutrumpėtų 25% ar šiek tiek daugiau.

3.1 apibrėžimas. *Kodavimas, leidžiantis gauti trumpesnį užkoduotą pranešimą nei pradinis pranešimas, vadinamas* duomenų spaudimu *arba* duomenų glaudinimu*. Spaudimo kokybė išreiškiama* spaudimo koeficientu*, kuris paprastai matuojamas procentais ir parodo, kiek procentų suspaustas pranešimas yra trumpesnis už pradinį.*

Šiuo metu duomenų spaudimas informacinėse technologijose yra plačiai naudojamas. Pavyzdžiui, spaudžiami failai (naudojami įvairūs formatai — zip, gz, rar,...), duomenys duomenų bazėse, modemais perduodami duomenys. Kosminiai laivai, siųsdami kitų planetų, palydovų ir t. t. nuotraukas į Žemę, taip pat naudoja duomenų spaudimą.

Praktikoje naudojamos suspaudimo programos (zip, rar ir kitos) tekstinius failus suspaudžia 70% ir daugiau. Tai reiškia, kad jos naudoja ne abėcėlinį kodavimą (abėcėlinis, kaip matėme, gali pasiekti tik šiek tiek daugiau nei 25%). Kokį kodavimą jos naudoja?

Idėja būtų tokia. Užkoduojamas ne kiekvienas simbolis atskirai, o simbolių seka. Pavyzdžiui, pranešimas skaidomas į žodžius (primenu, kad žodis — tai bet kokia simbolių seka, taigi nebūtinai mums įprasta prasme, kai laikome, kad žodžius vieną nuo kito skiria tarpai ir skyrybos ženklai), kiekvienas žodis laikomas naujos abėcėlės simboliu ir užkoduojamas, naudojant kurį nors abėcėlinį kodą tai naujai abėcėlei. Gautas žodžių ir jų kodų rinkinys vadinamas *žodynu*. Tada užkoduotas pranešimas bus sudarytas iš žodyno ir iš pradinio pranešimo žodžių kodų sekos. Dekoduojant tiesiog pakeisime kodus atitinkamais žodžiais iš žodyno.

3.2 pavyzdys. Tarkime, reikia koduoti lietuviškus tekstus. Skaidykime juos į žodžius pagal natūralias kalbos taisykles: žodžius vieną nuo kito skiria tarpai ir skyrybos ženklai. Galima daryti prielaidą, kad kiekviename konkrečiame tekste yra ne daugiau, kaip 2^{16} skirtingų žodžių (paprastai jų būna mažiau). Tokiu būdu, kiekvienam žodžiui galima priskirti numerį — sveikąjį skaičių iš 2 baitų (baitas yra 8 bitų seka). Kadangi vidutiniškai lietuviški žodžiai yra sudaryti daugiau, nei iš dviejų raidžių (o kiekviena raidė užrašoma vienu baitu), tai vietoj kiekvieno žodžio užrašę jo numerį iš 2 baitų, tekstą neblogai suspaudžiame (apie 65% normaliems lietuviškiems tekstams, nes vidutinis lietuviškų žodžių ilgis yra apie 6 raides). Aišku, dar turime priskaičiuoti ir žodyno dydį, nes jį reikia perduoti kartu su užkoduotu tekstu, bet jei pradinis tekstas buvo didelis (milijonai simbolių), tai prijungus žodyną, užkoduoto teksto ilgis padidėja palyginus nedaug. □

3.2 Lempelo–Zivo algoritmas

Praktikoje, kad nereikėtų kartu perduoti ir žodyno, naudojamas metodas, vadinamas *adaptyviuoju spaudimu*. Jo esmė tokia. Analizuojant pradinį tekstą, vienu metu dinamiškai sudarinėjamas žodynas ir koduojamas tekstas. Žodyno saugoti nereikia, nes dekoduojant jis vėl dinamiškai sudaromas iš užkoduoto teksto.

Panagrinėkime šios idėjos paprasčiausią realizaciją, vadinamą *Lempelo–Zivo algoritmu*[2]. Turime tekstą, kurį norime suspausti. Pradžioje žodyne yra tik tuščias žodis, pažymėtas numeriu 0. Tekstas skaidomas į žodžius taip. Tarkime, iki i-tojo simbolio tekstas jau suskaidytas. Einamasis žodis prasidės $i+1$-uoju simboliu ir baigsis j-uoju simboliu taip, kad tai būtų ilgiausias žodyne esantis žodis plius vienas simbolis, t. y. žodis nuo $i+1$-ojo simbolio iki $j-1$-ojo dar yra žodyne, o žodžio nuo $i+1$-ojo simbolio iki j-ojo jau nėra.

3.3 pavyzdys. Pavyzdžiui, jei užkoduojame tekstą, į kurį įeina žodžiai „diskrečioji matematika“, ir iki raidės „j“ (neįskaitant) jau suskaidėme tekstą į žodžius, tai einamasis žodis prasidės raide „j“. Jei žodyne, pavyzdžiui, jau buvo žodžiai „j“ ir „ji“, bet nebuvo žodžio „ji_“ (čia „_“ žymi tarpo simbolį), tai einamasis žodis bus „ji_“. Jei jau buvo žodis „ji mate“ ir visi jo priešdėliai, bet nebuvo žodžio „ji matem“, tai einamasis žodis bus „ji matem“. □

Prie užkoduoto teksto prijungiame žodyne rasto žodžio numerį ir tą papildomą simbolį, o einamąjį žodį dedame į žodyną. Ir kartojame šią procedūrą, kol užkoduojame visą tekstą. Taigi užkoduotas tekstas bus seka porų (p, q), kur p yra žodžio numeris žodyne, o q — simbolis. Baigus koduoti, sudarytą žodyną galime išmesti, kad jis neužimtų vietos, nes mes galėsime jį atstatyti dekodavimo metu.

Dekoduojame taip. Pradžioje žodyne yra tik tuščias žodis, pažymėtas numeriu 0. Imame einamąją porą (p, q) iš užkoduoto teksto. Imame p-tąjį žodyno žodį, prie jo prijungiame simbolį q ir gauname einamąjį žodį. Jį prijungiame prie dekoduojamo teksto ir įdedame į žodyną. Kartojame šią procedūrą, kol dekoduojame visą tekstą.

3.4 pavyzdys. Šis algoritmas labiau tinka ilgiems tekstams, nes juose raidžių kombinacijos pradeda kartotis. Kadangi čia galime duoti tik trumpą pavyzdį, imkime tokį tekstą, kuriame kai kurios raidžių kombinacijos kartojasi, pavyzdžiui, $S = oi\_oi\_ojojoi\_oi\_ojoi\_ojoi$, čia ženklu „_“ žymime tarpo simbolį. Užkoduokime šį pranešimą. Pradžioje žodyne D yra tik tuščias žodis, ir jo numeris yra 0.

- Išskirkime pirmą žodį duotame tekste. Jis prasidės pirma raide o. Žiūrime, ar žodis o yra žodyne. Nėra. Taigi ilgiausias žodyne esantis tinkamas žodis yra tuščias, todėl prie užkoduoto teksto C (kuris irgi pradžioje yra tuščias) prijungiame porą $(0, o)$, o į žodyną pirmu numeriu įdedame žodį o.

- Išskirkime antrą žodį. Jis prasidės antra raide: i, todėl analogiškai prie C prijungiame $(0, i)$, o į D antru numeriu įdedame i. Lygiai taip pat paskui prie C prijungiame $(0, \_)$, o į D trečiu numeriu įdedame _.

[2]Šis algoritmas paprastai vadinamas LZ78. Plačiau pasiskaityti apie šį algoritmą ir įvairius jo variantus galite Internete, pavyzdžiui, http://en.wikipedia.org/wiki/LZ78 arba http://oldwww.rasip.fer.hr/research/compress/algorithms/fund/lz/index.html

- Ketvirtas žodis vėl prasidės raide o. Ir ši raidė jau yra žodyne. Todėl einamasis žodis bus oi, kurio žodyne dar nėra: prie C prijungiame $(1, i)$, kur 1 yra žodžio o numeris žodyne, o i yra papildomas simbolis, o į D ketvirtu numeriu įdedame oi.
- Penktas žodis prasidės raide _, kuri yra žodyne, o $\_o$ jau nėra žodyne, todėl tai ir bus einamasis žodis: prie C prijungiame $(3, o)$, kur 3 yra žodžio _ numeris žodyne, o o yra papildomas simbolis, o į D penktu numeriu įdedame $\_o$.
- Taip tęsdami, gauname žodyną D ir užkoduotą pranešimą C, pateiktus 2 lentelėje. Paskutinėje C poroje nėra simbolio, nes baigėsi tekstas.

D	C
1 o	0 o
2 i	0 i
3 _	0 _
4 oi	1 i
5 $\_o$	3 o
6 j	0 j
7 oj	1 j
8 $oi\_$	4 _
9 oi_o	8 o
10 jo	6 o
11 $i\_$	2 _
12 ojo	7 o
	2

2 lentelė: Lempelo–Zivo algoritmo veikimo pavyzdys

Matome, kad šiame pavyzdyje užkoduotas pranešimas C nėra trumpesnis už pradinį pranešimą S. Taip yra dėl to, kad pavyzdys labai trumpas. Jei jis būtų ilgesnis, žodžius žodyne gautume ilgesnius, ir pakeitę juos jų numeriais labai sutrumpintume pranešimą.

Dabar žodyną D galima išmesti, kad neužimtų vietos, ir laikyti tik užkoduotą pranešimą

$$C = 0o0i0\_1i3o0j1j4\_8o6o2\_7o2.$$

Jį dekoduosime taip. Žodynas D vėl tuščias.

- Imame pirmą porą $(0, o)$ iš C. Einamąjį žodį gauname taip: 0-inį žodyno žodį (t. y. tuščią žodį) jungiame su raide o, gauname žodį o, kurį ir prijungiame prie sudarinėjamo pranešimo S bei įdedame į žodyną pirmu numeriu. Lygiai tą patį padarome su $(0, i)$ ir $(0, \_)$. Po šių operacijų $S = oi\_$ ir D turi 3 žodžius: o, i ir _.
- Tada imame porą $(1, i)$, taigi pirmą žodymo žodį (t. y. o) jungiame su raide i, gauname žodį oi, kurį ir prijungiame prie sudarinėjamo pranešimo S bei įdedame į žodyną ketvirtu numeriu.
- Taip tęsdami, gauname tą patį žodyną D, kaip ir užkoduodami, ir kartu dekoduojame užkoduotą pranešimą C. □

3.5 pastabos. 1. Praktikoje žodyno augimas yra ribojamas. Paprastai riba yra 2^{16} žodžių.

2. Praktikoje naudojami įvairūs šio algoritmo patobulinimai, pavyzdžiui, galima pradžioje imti ne tuščią žodyną, o jau su įdėtais ilgio 1 žodžiais, ir pan.

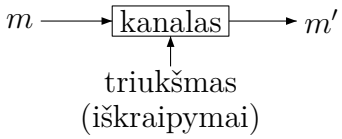
Iki šiol šnekėjome tik apie *spaudimą be informacijos praradimo*, t. y. pradinis pranešimas gali būti visiškai atgamintas iš užkoduoto pranešimo. Bet norint suspausti skaitmeninius vaizdus, garsą ar video signalą, gali būti prasminga kalbėti apie *spaudimą, prarandantį informaciją*, nes ši informacija paprastai skirta žmogui, ir galbūt ne visos pradinio pranešimo detalės yra būtinos ar net apskritai pajuntamos.

Vaizdams ir garsui suspausti be informacijos praradimo taikomos spėjamosios (prediktyvinės) schemos. Užkodavimo idėja tokia: remiantis iki šiol gautais duomenimis, numatyti kitos duomenų porcijos, pavyzdžiui, vaizdo taško ar garso imties, reikšmę, ir užkoduoti skirtumą tarp to spėjimo ir tikros reikšmės. Skirtumai paprastai būna maži, todėl čia gerai veikia Hafmeno kodavimas, mažas reikšmes užkoduojantis trumpais žodžiais. Dekoduojant daroma taip: norint gauti tikrą reikšmę, lygiai taip pat numatoma kita reikšmė, ir pridedamas dekoduotas skirtumas. Pavyzdžiui, pats paprasčiausias numatymo būdas — tarti, kad kita reikšmė bus lygiai tokia pati, kaip ir prieš tai buvusi. Gana gerai tinka spausti paprastai kompiuterinei grafikai ar juodai–baltam faksui. O vienas neprarandantis informacijos algoritmas iš JPEG vaizdų spaudimo algoritmų šeimos[3] numatymui naudoja aplinkinių taškų tiesines funkcijas, pavyzdžiui, viena iš naudojamų tiesinių funkcijų yra tokia: numatoma, kad spalvos reikšmė $I_{i,j}$ vaizdo taške (i, j) bus lygi $I_{i-1,j} + I_{i,j-1} - I_{i-1,j-1}$. Panašios schemos gerai veikia ir garsui, ir video signalui.

Bet vaizdui ir garsui suspausti dažnai užtenka kodavimo, prarandančio informaciją. Iš tikro, iki šiol mes darėme prielaidą, kad pradinis pranešimas yra skaitmeninio pavidalo ir turi būti perduotas be informacijos praradimo. Vaizdui ir garsui ši prielaida ne visada tinka, nes pradinis pranešimas gali būti analoginis, arba kad ir skaitmeninis, bet su didesne skiriamąja geba, negu mums reikia, pavyzdžiui, 16 bitų garsas kalbai įrašyti. Tokiais atvejais pirmas spaudimo žingsnis būtų sumažinti reikšmių, kurias gali įgyti kiekvienas pradinio pranešimo taškas, skaičių — tai vadinama *kvantavimu*. Pavyzdžiui, sumažinti 24 bitų spalvas iki 8 bitų, t. y. pasirinkti 256 spalvas iš 2^{24}. Praktikoje taikomi metodai suskaido pradinį pranešimą į dalis, pavyzdžiui, garsas gali būti skaidomas į dažnių juostas, vaizdas — į ryškumą ir spalvą. Tada kiekviena dalis kvantuojama atskirai, atsižvelgiant į įvairius faktorius, pavyzdžiui, į žmogaus galimybes pajausti tą dalį. Tai leidžia geriau suspausti, pasinaudojant, pavyzdžiui, tuo, kad žmonių kalbai įrašyti dažniai, aukštesni nei 7 KHz, yra nebūtini, arba kad žmogaus akis nepastebi mažų spalvos pasikeitimų, o tik ryškumo, ir pan. JPEG ir MPEG naudoja tokius spaudimo būdus vaizdui ir garsui.

4 Klaidas taisantys kodai

Panagrinėkime tokią schemą. Mes turime kažkokį pranešimą m ir norime jį kam nors perduoti. Perduodant pranešimą, galimas jo iškraipymas. Tai galima pavaizduoti grafiškai šitaip:



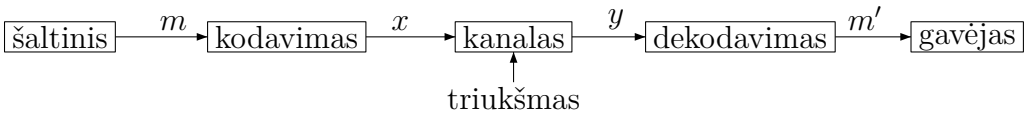
Pranešimas m perduodamas ryšio kanalu, kuriame jį gali iškraipyti triukšmas. Iš kanalo išeina pranešimas m', kuris gali skirtis nuo m ($m' \neq m$).

4.1 pavyzdys. Kanalų pavyzdžiai:

[3]Dažnai klaidingai manoma, kad JPEG yra kodavimas, prarandantis informaciją. Iš tikrųjų JPEG vardu vadinama standartų šeima, į kurią įeina ir keli neprarandantys informacijos algoritmai, žr., pavyzdžiui, http://en.wikipedia.org/wiki/Lossless_JPEG.

- Telefono linija. Informacija gali būti iškraipyta dėl triukšmo.
- Kosminis zondas siunčia Marso nuotraukas į Žemę.
- Ląstelės dalijimasis, kurio metu motininės ląstelės DNR perduoda informaciją dukterinės ląstelės DNR (perduodama informacija gali būti iškraipyta dėl radiacijos, ir tada įvyksta mutacija).
- Informacijos laikmena (pavyzdžiui, kietas diskas ar kompaktinė plokštelė): informacija į ją užrašoma, o po kurio laiko nuskaitoma. Per tą laiką jinai galėjo būti iškraipyta (dėl šilumos, radiacijos, subraižymų ir pan.) □

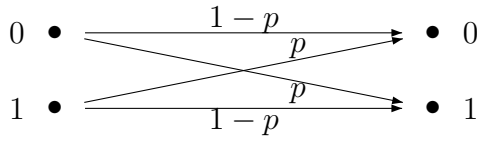
Norime, kad iš kanalo gautas pranešimas m' būtų lygus pradiniam pranešimui m su kuo didesne tikimybe. Kaip tai padaryti? Vienas kelias būtų gerinti kanalo charakteristikas, bet jis reikalauja daug lėšų. Klaidas taisantys kodai yra kitas sprendimas: priimame kanalą tokį, koks jis yra, bet perduodami juo informaciją, naudojame tam tikrus metodus, padedančius aptikti ir ištaisyti kanalo padarytas klaidas. Tai yra, pranešimas m prieš siunčiant į kanalą yra užkoduojamas, o gavus užkoduotą pranešimą iš kanalo, jis yra dekoduojamas. Schema būtų tokia:



Paprastai laikysime, kad m yra dvinaris (binarinis) vektorius $(m_1, m_2, \ldots, m_k)$, t. y. dvinarės abėcėlės $\mathcal{A} = \{0, 1\}$ vektorius. Paprastai mes jį rašysime kaip ilgio k žodį $m_1 m_2 \cdots m_k$. Prieš siųsdami į kanalą, jį užkoduojame dvinariu ilgio n žodžiu x, pridėdami papildomos informacijos, kuri leis aptikti ir ištaisyti klaidas. Taigi žodis x paprastai būna ilgesnis negu m, t. y. $n \geqslant k$. Išėjęs iš kanalo galbūt iškraipytas užkoduotas žodis y yra dekoduojamas, ir randamas žodis m'. Naudojant gerus kodavimo būdus, tikimybė, kad $m' \neq m$, labai sumažėja, bet užtat išauga simbolių kiekis, kurį reikia persiųsti kanalu.

Šiuo metu klaidas aptinkantys ir taisantys kodai yra plačiai naudojami informacinėse technologijose ir ryšiuose, pavyzdžiui, kompiuteriuose, modemuose, kompaktinėse plokštelėse, kosminiuose zonduose, mobiliuosiuose telefonuose ir taip toliau. Tvirtinama, jog dėl kompaktinėje plokštelėje naudojamų klaidas taisančių kodų muzikos kokybė teoriškai neturėtų nukentėti net ir pradūrus joje 2mm skersmens skylę!

Kanalą sumodeliuoti matematiškai galima įvairiais būdais. Mes naudosime vieną paprasčiausių modelių, vadinamą *dvinariu simetriniu kanalu* su klaidos tikimybe p. Laikysime, kad į kanalą siunčiami simboliai iš abėcėlės $\mathcal{A} = \{0, 1\}$, iš kanalo išeina irgi tos pačios abėcėlės simboliai, o kiekvieno simbolio iškraipymo tikimybė yra p, $0 \leqslant p < 0{,}5$. Grafiškai šį kanalo modelį galima pavaizduoti taip:



Matome, kad į kanalą gali įeiti simboliai 0 ir 1, išeina irgi 0 ir 1, simbolis 0 išlieka 0 su tikimybe $1 - p$ ir keičiasi į 1 su tikimybe p, analogiškai ir simbolis 1.

4.2 apibrėžimas. (Klaidas taisančiu) kodavimu *vadinsime injektyvų atvaizdį* $c : \mathcal{A}^k \to \mathcal{A}^n$. *Aibė* $C = c(\mathcal{A}^k) = \{c(m) : m \in \mathcal{A}^k\} \subseteq \mathcal{A}^n$ *vadinama* (klaidas taisančiu) kodu. *Kodo* C *vektoriai vadinami* kodo žodžiais. *Santykis* $R = k/n$ *vadinamas* kodo koeficientu.

Kodo koeficientas parodo, kuri į kanalą pasiųstų simbolių dalis yra naudinga informacija, o kuri tik pridėta klaidoms aptikti ir ištaisyti. Kuo jis didesnis, tuo geriau, nes tuo didesnė siunčiamo pranešimo dalis yra naudinga informacija.

4.3 pavyzdys. (Pakartojimo kodas) Tegu $k = 1$, $n = 3$. Taigi pranešimą dalijame į ilgio 1 žodžius ir kiekvieną tokį žodį užkoduojame ilgio 3 žodžiu taip: $c(0) = 000$, $c(1) = 111$. Tada kodas

$$C = \{000, 111\} \subset \{0, 1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\},$$

ir kodo koeficientas $R = 1/3$. □

Taigi visą informaciją, kurią norime išsiųsti, skaidome į k ilgio dvinarius vektorius (blokus) m, kiekvieną bloką užkoduojame n ilgio kodo žodžiu x (t. y. dvinariu vektoriumi iš kodo C) ir siunčiame į kanalą. Iš kanalo gauname iškraipytą vektorių y, kuris gali nebepriklausyti kodui C, t. y. y yra bet kuris vektorius iš aibės $\mathcal{A}^n$. Dekoduojame, vektoriui y priskirdami vektorių $m' \in \mathcal{A}^k$. Dekodavimas paprastai vyksta dviem etapais: pirmiausiai vektoriui y priskiriame kodo žodį $x' \in C$, o tada pasinaudojame kodavimo c atvirkštine funkcija $c^{-1} : C \to \mathcal{A}^k$, kad žodžiui $x' \in C$ priskirtume $m' \in \mathcal{A}^k$. Pirmame etape naudojama funkcija $f : \mathcal{A}^n \to C$ vadinama *dekodavimo taisykle*.

4.4 pavyzdys. Imkime tą patį pakartojimo kodą. Dekodavimas galėtų būti toks: kokių simbolių gautame žodyje daugiau, tokiu simboliu ir dekoduojame, pavyzdžiui, jei $y = 101$, tai jį dekoduojame 1, nes vektoriuje y yra du vienetai ir tik vienas nulis. Kitaip sakant, jei apibrėžtume atstumą tarp dviejų vektorių kaip skaičių koordinačių, kuriose jie skiriasi, tai dekoduotume imdami artimiausią (matuojant tuo atstumu) kodo žodį, ir tada imdami pranešimą, atitinkantį tą kodo žodį. Pvz, atstumas tarp 101 ir 000 yra du, tarp 101 ir 111 yra vienas, tai pasirenkame 111, nes jis arčiau 101, nei 000, ir dekoduojame 1, nes jis atitinka 111. □

Pavyzdyje įvestą atstumą ir dekodavimo procedūrą galime aprašyti formaliai.

4.5 apibrėžimas. Hemingo *(R.W. Hamming)* atstumu *(arba tiesiog* atstumu*) tarp dviejų vektorių* $x = (x_1, x_2, \ldots, x_n) \in \mathcal{A}^n$ *ir* $y = (y_1, y_2, \ldots, y_n) \in \mathcal{A}^n$, *žymimu* $d(x, y)$, *vadinsime koordinačių, kuriose jie skiriasi, skaičių:*

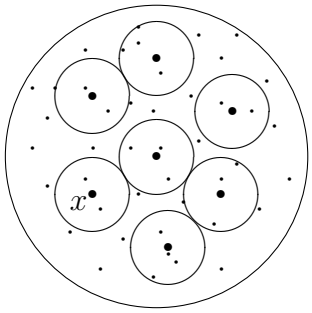
$$d(x, y) = |\{i : 1 \leqslant i \leqslant n,\ x_i \neq y_i\}|.$$

4.6 apibrėžimas. Minimalaus atstumo *dekodavimo taisykle vadinsime tokią dekodavimo taisyklę* $f : \mathcal{A}^n \to C$, *kuri kiekvienam* $y \in \mathcal{A}^n$ *priskiria arčiausiai (matuojant Hemingo atstumu) esantį kodo* C *žodį* x', *t. y. priskiria tokį* $x' \in C$, *kad*

$$d(y, x') = \min_{z \in C} d(y, z).$$

Šios taisyklės naudojimas grindžiamas taip. Kadangi klaidos tikimybė kanale yra mažesnė nei $1/2$, tai klaida yra mažiau tikėtina, nei klaidos nebuvimas. Lygiai taip pat dvi klaidos yra mažiau tikėtinos, nei viena, ir t. t. Taigi didžiausia tikimybė, kad į kanalą buvo išsiųstas kodo žodis, mažiausiai tesiskiriantis nuo iš kanalo gauto žodžio, nes tokiu atveju buvo padaryta mažiausiai klaidų. Todėl ir dekoduojame kodo žodžiu, mažiausiai tesiskiriančiu nuo iš kanalo gauto žodžio.

Dekodavimą naudojant minimalaus atstumo dekodavimo taisyklę grafiškai galima pavaizduoti taip:



Čia didysis skritulys su taškais yra aibė visų vektorių iš $\mathcal{A}^n$, pajuodinti taškai priklauso kodui $C \subseteq \mathcal{A}^n$. Iš kanalo gautas vektorius y gali būti bet kuris taškas. Pažiūrime, kuris kodo žodis (pajuodintas taškas) yra arčiausiai, juo ir dekoduojame. Pavyzdžiui, pasirinkime kurį nors kodo žodį x. Tarkime, artimiausias kitas kodo C žodis yra atstumu h nuo x. Tai reiškia, kad jei y yra nutolęs nuo x mažesniu atstumu, nei $h/2$, tai jis tikrai bus dekoduotas žodžiu x, nes šis kodo C žodis yra arčiausiai. Taigi apie x galime apibrėžti spindulio $h/2$ skritulį, ir visi jam priklausantys taškai bus dekoduojami to skritulio centru — kodo žodžiu x. Pažymėkime d patį mažiausią atstumą tarp bet kurių skirtingų kodo C žodžių, t. y.

$$d = \min_{x,z \in C,\ x \neq z} d(x, z).$$

Skaičius d vadinamas kodo C *minimaliu atstumu*. Tai jei apie kiekvieną kodo žodį apibrėšime spindulio $d/2$ skritulį, skritulyje esantys taškai visada bus dekoduojami skritulio centru.

Tarkime, į kanalą pasiuntėme žodį x, ir kanale jame buvo padaryta t klaidų. Iš kanalo išėjo vektorius y, esantis atstumu t nuo x. Jei $t < d/2$, tai y priklausys žodžio x skrituliui, todėl dekoduosime žodžiu x, t. y. ištaisysime kanalo padarytas klaidas. Jei $t > d/2$, tai y gali atsidurti jau kito kodo žodžio skritulyje, ir tokiu atveju bus dekoduotas neteisingai. Jei $t = d/2$, tai y gali priklausyti dviem skrituliams (gali būti ant dviejų skritulių krašto), ir nežinosime, kurio rutulio centru dekoduoti. Jei y neatsiduria jokio kodo žodžio skritulyje, tai, jį dekoduodami artimiausiu kodo žodžiu, galime dekoduoti teisingai, o galime ir klaidingai. Taigi, jei $t \geqslant d/2$, nesame garantuoti dėl dekodavimo teisingumo.

4.7 apibrėžimas. *Jei, naudojant minimalaus atstumo dekodavimo taisyklę, dekoduojama visada teisingai, kai siųstame žodyje yra ne daugiau kaip t klaidų, tai kodą C vadiname* t klaidų taisančiu kodu.

t klaidų taisantį kodą vadiname tiksliai t klaidų taisančiu kodu*, jei jis nėra $t+1$ klaidą taisantis kodas.*

Tegu $[a]$ yra skaičiaus a sveikoji dalis, t. y. didžiausias sveikasis skaičius, mažesnis arba lygus už a.

4.8 teorema. *Kodas C yra tiksliai $[(d-1)/2]$ klaidų taisantis kodas, kur d yra kodo C minimalus atstumas.*

Įrodymas. Kaip matėme, kodas C yra t klaidų taisantis kodas, jei $t < d/2$. Dėl to jis yra tiksliai t klaidų taisantis kodas, jei t yra didžiausias sveikasis skaičius, mažesnis už $d/2$. Jei d yra lyginis, tai didžiausias sveikasis skaičius, mažesnis už $d/2$, yra $d/2 - 1 = (d-2)/2 = [(d-1)/2]$. Jei d yra nelyginis, tai didžiausias sveikasis skaičius, mažesnis už $d/2$, yra $d/2 - 1/2 = (d-1)/2 = [(d-1)/2]$. Abiem atvejais gauname pageidaujamą rezultatą. □

Todėl stengiamasi sudaryti tokius kodus, kurių minimalus atstumas d būtų kuo didesnis, kad kodas ištaisytų kuo daugiau klaidų.

Kartais svarbu ne tik tai, kiek klaidų kodas ištaiso, bet ir kiek jų aptinka. Kadangi į kanalą siunčiame tik kodo žodžius, tai, jei iš kanalo išeina ne kodo žodis, reiškia, buvo padaryta klaidų. Todėl t klaidų aptinkantis kodas gali būti apibrėžiamas taip:

4.9 apibrėžimas. *Kodą C vadiname t* klaidų aptinkančiu kodu*, jei bet kuriame kodo žodyje įvykus ne daugiau kaip t klaidų, gautas vektorius nepriklauso kodui C.*

t klaidų aptinkantį kodą vadiname tiksliai t klaidų aptinkančiu kodu*, jei jis nėra $t+1$ klaidą aptinkantis kodas.*

Jei įvyko d iškraipymų, tai gali būti, kad gavome kitą kodo žodį, todėl kodas C yra t klaidų aptinkantis kodas, jei $t < d$. Gauname tokį rezultatą:

4.10 teorema. *Kodas C yra tiksliai $d-1$ klaidą aptinkantis kodas, kur d yra kodo C minimalus atstumas.*

Panagrinėkime kelis paprastų kodų pavyzdžius.

4.11 pavyzdžiai. 1. *Pakartojimo n kartų kodas.* Tai prieš tai buvusio pavyzdžio apibendrinimas. Čia $k = 1$, o n — kažkoks fiksuotas natūralusis skaičius. Ilgio 1 žodžius užkoduojame ilgio n žodžiais: $c(0) = 00\cdots0$, $c(1) = 11\cdots1$. Kodas $C = \{00\cdots0, 11\cdots1\} \subseteq \{0,1\}^n$. Kodo koeficientas $R = 1/n$ — labai žemas, užtat kodas, kaip matysime, ištaiso ir aptinka daug klaidų. Labai paprasta rasti šio kodo minimalų atstumą — iškart matome, kad $d = n$. Taigi tai tiksliai $[(n-1)/2]$ klaidų taisantis ir tiksliai $n-1$ klaidą aptinkantis kodas. Dekodavimas vyksta taip: gavę iš kanalo vektorių y, suskaičiuojame, ko daugiau jame yra — nulių ar vienetų, tuo ir dekoduojame. Jei n lyginis, gali būti, kad vienetų ir nulių bus po lygiai. Tokiam atvejui turime įsivesti kažkokią atskirą taisyklę, pavyzdžiui, atsitiktinai pasirinkti 0 ar 1, arba visada dekoduoti 0, ir pan.

2. *Kontrolinio simbolio kodas.* Kartais svarbu ne ištaisyti klaidas, o labai nesunkiai jas aptikti. Šis kodas tam ir skirtas. Pranešimą $m = (m_1, m_2, \ldots, m_k)$ užkoduojame, prijungdami vieną papildomą simbolį x_{k+1}, vadinamą *kontroliniu simboliu*, t. y. užkoduotas pranešimas $x = (m_1, m_2, \ldots, m_k, x_{k+1})$. Taigi čia k yra kažkoks fiksuotas natūralusis skaičius, o $n = k+1$. Kontrolinis simbolis apskaičiuojamas taip, kad užkoduotame vektoriuje būtų lyginis vienetų skaičius. Pavyzdžiui, jei $m = 1001$, tai $x = 10010$, o jei $m = 1011$, tai $x = 10111$. Kodas C yra sudarytas iš visų ilgio n dvinarių vektorių, turinčių lyginį vienetų skaičių. Kodo koeficientas $R = k/(k+1)$ yra labai aukštas, artimas vienetui, užtat kodas, kaip matysime, klaidų visai netaiso, ir aptinka tik vieną klaidą. Minimalus atstumas irgi nesunkiai randamas — tai $d = 2$. Iš tikro, atstumas tarp kodo žodžių $000\cdots0$ ir $110\cdots0$ yra 2, ir jokie du kodo žodžiai nėra nutolę atstumu 1 vienas nuo kito (jei taip būtų, viename iš jų vienetų skaičius būtų nelyginis). Todėl šis kodas taiso $[(d-1)/2] = 0$ klaidų, ir aptinka $d-1 = 1$ klaidą. Tai yra, jei padaryta viena klaida, tai kodas būtinai ją aptiks, o jei daugiau, tai gali ir nebeaptikti. Bet gali ir aptikti. Pavyzdžiui, šis kodas aptiks, kad padaryta klaidų, jei jų skaičius nelyginis. Dekoduodami tiesiog patikriname, ar vienetų skaičius gautame vektoriuje lyginis. Jei taip, tai dekoduotas vektorius bus gautas, atmetus paskutinę koordinatę. Jei ne, reiškia, buvo klaidų. Teks paprašyti persiųsti iš naujo.

3. *Knygų numeracijos sistema ISBN.* Tai praktikoje taikomo kontrolinio simbolio kodo pavyzdys. Kiekviena šiais laikais išleidžiama knyga turi ISBN (International Standard Book Number) numerį, sudarytą iš dešimties dešimtainių skaitmenų. Pavyzdžiui, V.Stakėno knygutės „Informacijos kodavimas“ ISBN numeris yra 9986-19-183-1. Pirmi devyni skaitmenys $a_1, a_2, \ldots, a_9$ rodo šalį, kurioje išleista knyga, leidyklą ir knygos numerį, o dešimtas a_{10} yra kontrolinis, apskaičiuojamas pagal tokią formulę:

$$a_{10} \equiv \sum_{i=1}^{9} i a_i \pmod{11}.$$

(t. y. apskaičiuojame sumą, dalijame ją iš 11 ir imame liekaną). Dalijant iš 11, liekana gali būti 10. Tokiu atveju kontrolinis simbolis a_{10} žymimas ženklu X. Taigi $k = 9$, $n = 10$, $R = 9/10$. Šis kodas klaidų netaiso, bet aptinka pavienes klaidas ir dviejų greta stovinčių simbolių sukeitimą vietomis (tai dažniausiai pasitaikančios klaidos suvedant knygos ISBN numerį). Dekodavimas yra lygiai toks pat, kaip ir kodavimas: apskaičiuojame tą pačią liekaną ir palyginame su a_{10}. Jei nelygu, reiškia, įvyko klaida, reikia prašyti atsiųsti ISBN numerį iš naujo.

4. *Lietuvos piliečių asmens kodas.* Kiekvienas Lietuvos pilietis turi savo asmens kodą, kuriame irgi naudojamas kontrolinis simbolis. Asmens kodo struktūra:

$$LY_1Y_2M_1M_2D_1D_2X_1X_2X_3K,$$

kur

L — lytis. Lietuviai turi 6 lytis:

1 — vyras, gimęs XIX amžiuje,
2 — moteris, gimusi XIX amžiuje,
3 — vyras, gimęs XX amžiuje,
4 — moteris, gimusi XX amžiuje,
5 — vyras, gimęs XXI amžiuje,
6 — moteris, gimusi XXI amžiuje.

$Y_1Y_2M_1M_2D_1D_2$ — gimimo data:

Y_1Y_2 — metai šimtmetyje,
M_1M_2 — mėnuo,
D_1D_2 — diena.

$X_1X_2X_3$ — eilės numeris tarp tą dieną gimusiųjų, priskiriamas Gyventojų registro.

K — kontrolinis skaičius, apskaičiuojamas dauginant kiekvieną asmens kodo skaitmenį iš svorio koeficiento ir sumuojant:

$$S = L \cdot 1 + Y_1 \cdot 2 + Y_2 \cdot 3 + M_1 \cdot 4 + M_2 \cdot 5 + D_1 \cdot 6 + D_2 \cdot 7 + X_1 \cdot 8 + X_2 \cdot 9 + X_3 \cdot 1.$$

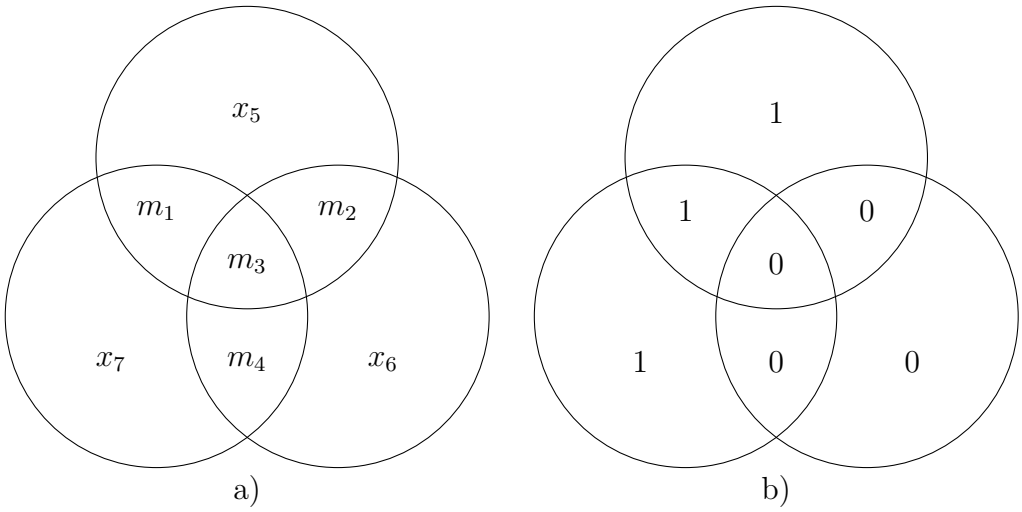
Suma S dalijama iš 11, ir jei liekana nelygi 10, ji yra asmens kodo kontrolinis skaičius.

Jei liekana lygi 10, tuomet skaičiuojama nauja suma su tokiais svorio koeficientais:

$$S = L \cdot 3 + Y_1 \cdot 4 + Y_2 \cdot 5 + M_1 \cdot 6 + M_2 \cdot 7 + D_1 \cdot 8 + D_2 \cdot 9 + X_1 \cdot 1 + X_2 \cdot 2 + X_3 \cdot 3.$$

Dar kartą dalijame iš 11, ir jei liekana nelygi 10, ji yra asmens kodo kontrolinis skaičius. Jei vėl liekana yra 10, kontrolinis skaičius imamas lygiu 0.

5. *Hemingo kodas.* Yra ištisa šeima dvinarių Hemingo kodų. Mes nagrinėsime tik vieną iš jų, kurio parametrai yra tokie: $k = 4$, $n = 7$, $R = 4/7$. Pranešimą $m = (m_1, m_2, m_3, m_4)$ užkoduojame žodžiu $x = (m_1, m_2, m_3, m_4, x_5, x_6, x_7)$. Užkodavimą galima pavaizduoti grafiškai taip. Surašome šiuos septynis simbolius į susikertančius skritulius taip, kaip parodyta 1a) paveiksle. Simboliai x_5, x_6, x_7 parenkami taip, kad kiekviename skritulyje esančių vienetų skaičius būtų lyginis. Pavyzdžiui, jei $m = (1, 0, 0, 0)$, tai $x = (1, 0, 0, 0, 1, 0, 1)$ (žr. 1b) pav.).



1 pav.: Kodavimas Hemingo kodu

Dekoduojame tokiu būdu. Parodysime, kad šis kodas visada ištaiso pavienę klaidą. Tarkime, klaida įvyko pozicijoje, kuri priklauso tik vienam skrituliui, pavyzdžiui, pozicijoje x_5. Dekoduodami patikriname, kuriuose skrituliuose vienetų skaičius yra nelyginis. Matome, kad viršutiniame skritulyje jis nelyginis, kituose — lyginis. Padarome išvadą, kad klaida įvyko pozicijoje, kuri priklauso viršutiniam skrituliui ir nepriklauso kitiems skrituliams. Tokia pozicija tėra tik viena, ir tai būtent x_5, galime ištaisyti joje padarytą klaidą. Lygiai taip pat vienareikšmiškai nustatome klaidos poziciją, jei klaida padaryta simbolyje, priklausiančiame lygiai dviem iš trijų skritulių, pavyzdžiui, m_2 (randame, kad vienetų skaičius nelyginis viršutiniame ir dešiniajame skrituliuose, ir nusprendžiame, kad klaida padaryta pozicijoje, kuri priklauso šiems dviem skrituliams, ir nepriklauso trečiajam), ir jei klaida padaryta simbolyje, priklausančiame visiems trims skrituliams. Taigi, kad ir kur būtų padaryta klaida, mes galime rasti jos poziciją ir ją ištaisyti.

Bet dviejų klaidų kodas jau nebeištaiso. Pavyzdžiui, tarkime, klaidos įvyko pozicijose m_4 ir x_7. Tada vienetų skaičius nelyginis pasidarys tik dešiniajame skritulyje, dėl to ištaisysime x_6: dekoduodami ne tik neištaisysime klaidų, bet dar daugiau jų įvelsime. Žodžiu, šis kodas yra tiksliai vieną klaidą taisantis kodas. Ir tiksliai 2 klaidas aptinkantis, nes padarius dvi klaidas,

būtinai kuriame nors skritulyje vienetų skaičius pasidarys nelyginis, iš ko ir nuspręsime, kad įvyko klaida, o įvykus trims klaidoms, vienetų skaičius visuose skrituliuose gali išlikti lyginis, ir klaidų galime nepastebėti (taip bus, pavyzdžiui, jei klaidos įvyks m_1, x_5 ir x_7 pozicijose).

Tai mums leidžia rasti dar vieną kodo parametrą: minimalus šio kodo atstumas $d = 3$. □

Baigiamosios pastabos. Daviau tik mažiukus pavyzdukus ir pačią teorijos pradžią, o iš tikro klaidas taisančių kodų teorija yra didžiulė ir šiuo metu sparčiai besivystanti. Tai viena iš kelių matematikos sričių, kur naujausios matematinės teorijos turi tiesioginį praktinį pritaikymą. Ir atvirkščiai, tai pavyzdys, kaip iš praktinio uždavinio išsivystė graži matematinė teorija.

5 Kriptografija

5.1 Įvadas

Įsivaizduokite tokią situaciją: du asmenys — A ir B (vadinkime juos Algiu ir Birute) — nori pasikeisti slapta informacija, o trečias asmuo Z (Zigmas) nori ją sužinoti. Kad Zigmas jos nesužinotų, perduodama informacija yra šifruojama. *Šifravimas* — tai duomenų kodavimas, norint paslėpti jų turinį. Užšifruotas pranešimas vadinamas *šifru*. *Kriptografija* yra mokslas, kuriantis ir nagrinėjantis įvairias šifravimo sistemas, dar vadinamas *kriptografinėmis sistemomis, kriptosistemomis* arba *šifrais*.

Įvairios kriptosistemos naudojamos jau nuo antikos laikų. Jau minėjau Cezario šifrą, kai lotynų abėcėlės raidė užšifruojama raide, esančia trimis pozicijomis abėcėlėje toliau, t. y. raidė A užšifruojama raide D, raidė B - raide E, ir t. t. Tokios paprastos kriptosistemos pasižymi tuo, kad norint išsaugoti pranešimo slaptumą, reikia slėpti patį šifravimo algoritmą. Kai Zigmas algoritmą sužinos, reikės galvoti kitą algoritmą, o tai tikrai nelengva. Geriau būtų, jei šifravimo algoritmas turėtų parametrą, vadinamą *raktu* (pavyzdžiui, tai gali būti skaičius), kurį ir reikėtų slėpti. Jei Zigmas jį sužinos, pakeisime jį kitu raktu, ir Zigmas vėl negalės dešifruoti mūsų pranešimų, nors ir naudosime tą patį algoritmą.

Galima apibendrinti Cezario šifrą taip, kad jis priklausytų nuo rakto s: užšifruojamą raidę mes „pastumiame“ per s pozicijų abėcėlėje (Cezario šifro atveju $s = 3$). Zigmui sužinojus šį raktą, mes galime pasirinkti kitą. Bėda ta, kad šiuo atveju galimų raktų aibė yra labai maža (raktų yra tiek, kiek abėcėlėje raidžių), ir Zigmas, žinodamas šifravimo algoritmą, perrinkimo būdu ras panaudotą raktą. Matome, kad kriptosistemos raktų aibė turi būti pakankamai didelė, kad nebūtų galima jos perrinkti.

Galime toliau apibendrinti šią sistemą: perstumkime raides ne per tris pozicijas, o per kintamą pozicijų skaičių, priklausantį nuo rakto. Tarkime, raktas yra žodis `aibė`, jo raidės yra atitinkamai 1-oji, 13-oji, 3-ioji ir 9-oji lietuviškos abėcėlės raidės. Norėdami užšifruoti pranešimą, pirmą jo raidę perstumiame per vieną poziciją, antrą per trylika, trečią per tris, ketvirtą per devynias, penktą vėl per vieną ir t. t. Nors Zigmas ir žinotų, kad Algis ir Birutė naudoja tokį šifravimo algoritmą, norėdamas dešifruoti, turėtų kažkokiu būdu sužinoti ir raktą, o tokių raktų aibė yra labai didelė.

Antrojo pasaulinio karo metu tokios rūšies šifravimo algoritmai, tik, aišku, žymiai sudėtingesni, ir buvo naudojami (garsiausias jų pavyzdys yra vokiečių ENIGMA). Norint dešifruoti priešininko pranešimus, buvo kuriamos sudėtingos mašinos. Jos ir tapo šiuolaikinių kompiuterių pradininkėmis. Beje, laikoma, kad sąjungininkų pergalę Antrajame pasauliniame kare iš dalies nulėmė tai, kad jie sugebėjo įveikti vokiečių šifrus.

Tokios kriptosistemos išsivystė į tokias dabar plačiai naudojamas kriptosistemas, kaip DES, IDEA ir kt. Jos vadinamos *slapto rakto* kriptosistemomis. Jos pasižymi tuo, kad ir šifravimui, ir dešifravimui naudojamas tas pats raktas (kuris dėl to turi būti slaptas, nes jei Zigmas jį sužinotų, irgi galėtų dešifruoti Algio Birutei siunčiamus pranešimus). Geriausiai žinoma slaptojo rakto kriptosistema yra DES. Šifruojant šia kriptosistema, pranešimo simboliai yra daug kartų keičiami vietomis, sudėdinėjami tarpusavyje ir su rakto simboliais, ir pan. Tobulėjant kompiuteriams, ši sistema darosi jau nebe tokia saugi, ir ją jau keičia naujas slapto rakto kriptosistemų standartas, vadinamas AES.

Didžiausias slaptojo rakto kriptosistemų trūkumas yra tai, kad raktas turi būti slaptas. Jei slaptai bendrauti tarpusavyje nori tūkstančiai žmonių, tai kiekviena jų pora turės turėti savo slaptą raktą. Kaip tvarkyti tiek raktų, ir, dar svarbiau, kaip jais apsikeisti, kad raktas išliktų slaptas? 1976 metais Diffie ir Hellman pasiūlė visiškai naujo tipo kriptosistemą, kuri išsprendžia šias problemas. Tokio tipo kriptosistemos vadinamos *viešo rakto* kriptosistemomis. Jos pasižymi tuo, kad turi vieną raktą šifravimui, o kitą dešifravimui. Raktas šifravimui gali būti viešas, prieinamas visiems, ir jis vadinamas *viešuoju* raktu. Raktas dešifravimui turi būti slaptas, kad tik jį žinantis galėtų dešifruoti pranešimą. Jis vadinamas *privačiuoju* raktu. Taigi Birutė, norėdama gauti šifruotus pranešimus, susigeneruoja sau raktų porą — viešąjį ir privatųjį raktus. Viešąjį paskelbia visiems, ir Algis tuo Birutės viešu raktu šifruoja pranešimus jai. Birutė, gavusi pranešimą, jį dešifruoja savo privačiuoju raktu. Kadangi to rakto daugiau niekas nežino, niekas kitas negali to pranešimo dešifruoti, tik Birutė.

Aišku, viešo rakto kriptosistemos turi būti tokios, kad viešojo rakto žinojimas neleistų rasti privačiojo rakto ir dešifruoti pranešimo. Paprastai viešo rakto kriptosistema sudaroma, naudojant funkciją f, kurios reikšmės yra lengvai apskaičiuojamos, bet jos atvirkštinės funkcijos f^{-1} reikšmės yra labai sunkiai apskaičiuojamos, nebent žinotum kai ką daugiau apie tą funkciją, kažkokį jos parametrą. Tada pranešimas m užšifruojamas, paprasčiausiai apskaičiuojant $f(m)$ reikšmę, o turėdamas $f(m)$, pradinį pranešimą $m = f^{-1}(f(m))$ gali apskaičiuoti tik tas, kuris žino kažkokios papildomos slaptos informacijos apie atvirkštinę funkciją f^{-1}, nes bendru atveju jos reikšmės yra labai sunkiai apskaičiuojamos. Ta papildoma slapta informacija apie f^{-1} ir yra privatus raktas.

5.2 RSA kriptosistema

Geriausiai žinoma viešo rakto kriptosistema yra RSA (pavadinta pagal jos kūrėjus — R. Rivest, A. Shamir, L. Adleman), pasiūlyta 1978 metais. Trumpai pasiaiškinkime, kokią funkciją jinai naudoja.

Visų pirma apibrėžkime vieną pagalbinę funkciją — skaičių teorijoje gerai žinomą *Oilerio funkciją* $\varphi(n)$. Tai skaičius tokių m, $1 \leqslant m < n$, kurie yra tarpusavyje pirminiai[4] su n, t. y.

$$\varphi(n) = |\{m : 1 \leqslant m < n,\ (m, n) = 1\}|,$$

kur (m, n) yra skaičių m ir n bendras didžiausias daliklis. Pavyzdžiui, $\varphi(6) = 2$, nes yra du skaičiai (1 ir 5), kurie yra tarpusavyje pirminiai su 6, o visi kiti (2,3 ir 4) turi su 6 bendrų daliklių. Apie Oilerio funkciją mums užteks žinoti tokį faktą: jei p ir q yra du skirtingi pirminiai skaičiai, tai $\varphi(pq) = (p-1)(q-1)$. Taigi, pagal šią formulę, turėtų būti $\varphi(6) = \varphi(2 \times 3) = (2-1)(3-1) = 2$ — ką mes ir esame gavę.

[4]Skaičiai a ir b yra *tarpusavyje pirminiai*, jei jie neturi bendrų daliklių, didesnių už 1.

Raktų parinkimas. Birutė pasirenka du skirtingus pirminius skaičius p, q ir juos sudaugina: $n = pq$. Dar jinai pasirenka tokį natūralųjį skaičių e, $1 < e < \varphi(n)$, kad jis būtų tarpusavyje pirminis su $\varphi(n) = (p-1)(q-1)$. Viešas raktas ir bus pora $K_v = (e, n)$. Privatus raktas bus pora $K_p = (d, n)$, kur d, $1 < d < \varphi(n)$, yra toks skaičius, kad $ed \equiv 1 \bmod \varphi(n)$. Primenu, kad $a \equiv b \bmod c$ reiškia, kad padaliję a iš c gausime tą pačią liekaną, kaip ir padaliję b iš c. Lengva pastebėti, kad pastaroji sąlyga yra ekvivalenti sąlygai $a-b \equiv 0 \bmod c$, t. y. skirtumą $a-b$ padalijus iš c, liekana bus 0, kitaip sakant, $a-b$ dalijasi iš c be liekanos, o tai reiškia, kad egzistuoja sveikasis skaičius t toks, kad $a - b = tc$, arba kitaip $a = b + tc$.

Tokį skaičių d rasti galima naudojantis Euklido algoritmu dviejų skaičių bendram didžiausiam dalikliui rasti, kurio čia nenagrinėsime. Skaičių p ir q daugiau nebereikės, ir juos galima ištrinti (ir netgi patartina, kad jie nepakliūtų į rankas Zigmui).

Šifravimas. Pranešimai, kuriuos bus galima siųsti Birutei, yra skaičiai nuo 2 iki $n-1$. Algis, turėdamas Birutės viešą raktą (e, n), užšifruoja pranešimą m tokiu būdu: $c \equiv m^e \bmod n$, $2 \leqslant c < n$ (apskaičiuoja m^e, dalija rezultatą iš n ir ima liekaną), ir siunčia užšifruotą pranešimą c Birutei.

Dešifravimas. Dešifravimo algoritmas visiškai toks pat kaip ir šifravimo, tik naudojamas privatus raktas: dešifruotas pranešimas r bus gaunamas tokiu būdu: $r \equiv c^d \bmod n$. Galima parodyti, kad r sutampa su m.

Įsitikinkime, kad dešifruotas pranešimas r sutampa su pradiniu pranešimu m. Parodykime, kad $r \equiv m \bmod n$. Iš pradžių parodykime, kad $r \equiv m \bmod p$. Jei $m \equiv 0 \bmod p$, tai $c \equiv 0 \bmod p$, todėl $r \equiv m \bmod p$. Tegu $(m, p) = 1$. Žinome, kad $c^d \equiv (m^e)^d \equiv m^{ed} \bmod p$. Be to, $ed \equiv 1 \bmod \varphi(n)$. Ši sąlyga reiškia, kad egzistuoja sveikasis skaičius t toks, kad $ed = 1 + t\varphi(n) = 1 + t(p-1)(q-1)$. Įstatę šią lygybę, gauname $m^{ed} \equiv m^{1+t(p-1)(q-1)} \equiv m\big(m^{p-1}\big)^{t(q-1)} \bmod p$. Dabar galime pasinaudoti mažąja Ferma teorema, kuri tvirtina, kad jei p — pirminis skaičius, tai su bet kokiu sveikuoju skaičiumi a, kuriam $(a, p) = 1$, $a^{p-1} \equiv 1 \bmod p$. Taigi gauname $r \equiv m\big(m^{p-1}\big)^{t(q-1)} \equiv m \cdot 1^{t(q-1)} \equiv m \bmod p$. Lygiai taip pat gauname, kad $r \equiv m \bmod q$. Todėl $r \equiv m \bmod n$. Be to, žinome, kad $m < n$ ir $r < n$, todėl $r = m$, ką ir reikėjo įrodyti.

Tarkime, jog Zigmas sužinojo, kad užkoduotas pranešimas yra c. Jis taip pat žino Birutės viešą raktą (e, n). Norėdamas rasti pradinį pranešimą m, jis turi išspręsti tokią lygtį: $m^e \equiv c \bmod n$. Pasirodo, tai padaryti yra labai sunku (aišku, pakankamai dideliam n). Jis gali bandyti ir kitaip: iš lygties $ed \equiv 1 \bmod \varphi(n)$ rasti privatų raktą d. Bet $\varphi(n) = (p-1)(q-1)$ jis taip pat neturi. Kad galėtų apskaičiuoti $\varphi(n)$, jis turi iš pradžių rasti p ir q, t. y. jis turi išskaidyti n pirminiais dauginamaisiais. Dideliems n tai irgi labai sunkus uždavinys.

Kad RSA kriptosistema būtų saugi, Birutės pasirinkti pirminiai skaičiai p ir q turi būti labai dideli, apie 200 dešimtainių skaitmenų. Dabar taikomi algoritmai leidžia greitai patikrinti, ar tokios eilės skaičius yra pirminis, ar ne (tai užtrunka tik kelias minutes). Taigi Birutei nekils problemų parenkant p ir q. O Zigmui kils, jei jis bandys skaidyti n pirminiais dauginamaisiais. Nes n bus sudarytas iš apie 400 skaitmenų, o dabartiniai algoritmai tokios eilės skaičiui išskaidyti sugaištų daugybę metų.

5.1 pavyzdys. *Raktų parinkimas.* Visų pirma turime parinkti raktus. Be abejo, mes negalime pasirinkti tokių didelių skaičių p ir q, kokie turėtų būti iš tikrųjų, bet tam, kad pamatytume, kaip veikia RSA, užteks ir nedidelių skaičių. Imkime, pavyzdžiui, $p = 7$, $q = 11$. Tada $n = 77$. Dabar reikia pasirinkti e, tarpusavyje pirminį su $\varphi(n) = (p-1)(q-1) = 6 \cdot 10 = 60$. Imkime, pavyzdžiui, $e = 13$. Taigi viešas raktas bus pora $(13, 77)$. Privatus raktas bus pora $(d, 77)$, kur d yra toks, kad $ed \equiv 1 \bmod \varphi(n)$, t. y. $13d \equiv 1 \bmod 60$. Galite nesunkiai patikrinti, kad $d = 37$ (iš tikro, $ed = 13 \cdot 37 = 481 = 1 + 8 \cdot 60$).

Šifravimas. Tarkime, kažkas nori mums perduoti pranešimą $m = 2$. Jisai užšifruoja šį pranešimą, pasinaudodamas mūsų viešu raktu: $2^{13} = 8192 \equiv 30 \bmod 77$ (dalija 2^{13} iš 77 ir ima liekaną). Užšifruotas pranešimas $c = 30$.

Dešifravimas. Kad dešifruotume pranešimą, mums reikia apskaičiuoti $30^{37} \bmod 77$. Pasinaudosime modulio savybe: jei $a' \equiv b' \bmod c$ ir $a'' \equiv b'' \bmod c$, tai $a'a'' \equiv b'b'' \bmod c$. Taigi, jei $a' \equiv b' \bmod c$, tai $a'^2 \equiv b'^2 \bmod c$. Gauname tokią skaičiavimų moduliu 77 seką:

$$\begin{aligned} 30^1 &= 30 \\ 30^2 &= 900 \equiv 53 \\ 30^4 &\equiv 53^2 = 2809 \equiv 37 \\ 30^8 &\equiv 37^2 = 1369 \equiv 60 \\ 30^{16} &\equiv 60^2 = 3600 \equiv 58 \\ 30^{32} &\equiv 58^2 = 3364 \equiv 53. \end{aligned}$$

Taigi

$$30^{37} = 30^{32} \cdot 30^4 \cdot 30^1 \equiv 53 \cdot 37 \cdot 30 = 1961 \cdot 30 \equiv 36 \cdot 30 = 1080 \equiv 2 \pmod{77}.$$

Gavome pranešimą, kuris ir buvo užšifruotas.

Zigmas. Ką gali padaryti Zigmas? Jei jam pasiseka gauti užšifruotą pranešimą $c = 30$, jisai turi rasti tokį m, kad $m^{13} \equiv 30 \bmod 77$. Šiuo metu nėra žinoma metodų, kurie būtų žymiai greitesni už paprasčiausią galimų reikšmių perrinkimą. Taigi Zigmas bandytų visus m nuo 2 iki $n - 1$ įstatyti ir tikrinti, ar jie tenkina lygybę. Kai n labai didelis, jis užgaištų tam labai ilgai.

Kitas kelias, kurį gali bandyti Zigmas: išskaidyti $n = 77$. Kaip matėme, dideliems n tai irgi užima labai daug laiko. □

5.3 Elektroninis parašas

Kartais svarbu ne pranešimo slaptumas, o jo tapatumas. Tokiu atveju naudojamas elektroninis parašas (dar vadinamas skaitmeniniu parašu).

Tarkime, Algis siunčia pranešimą Birutei. Prie siunčiamo pranešimo pridedamas kažkoks duomenų kiekis, kuris vadinamas *elektroniniu parašu*. Jisai užtikrina, kad pranešimą tikrai pasiuntė Algis, o ne koks nors kitas asmuo, ir kad pranešimas pakeliui nebuvo Zigmo pakeistas.

Turint viešo rakto kriptosistemą, galima sudaryti elektroninio parašo schemą. Imkime, pavyzdžiui, RSA.

Algis turi savo viešą raktą $K_v = (e, n)$ ir privatų $K_p = (d, n)$. Jisai nori pasiųsti pranešimą m Birutei. Prieš siųsdamas jis jį užšifruoja savo privačiu raktu (pasirašo), t. y. apskaičiuoja $c \equiv m^d \bmod n$, ir siunčia Birutei abu — ir pranešimą m, ir elektroninį parašą c. Birutė, norėdama įsitikinti, kad pranešimą m atsiuntė tikrai Algis, dešifruoja c pasinaudodama Algio viešu raktu, t. y. apskaičiuoja $r \equiv c^e \bmod n$, ir patikrina, ar $r = m$. Jei taip, tai tikrai siuntė Algis, nes tik jis žino savo privatų raktą d, todėl tik jis galėjo apskaičiuoti $c \equiv m^d \bmod n$. Ir pranešimas nebuvo iškraipytas, nes r ir m sutampa.