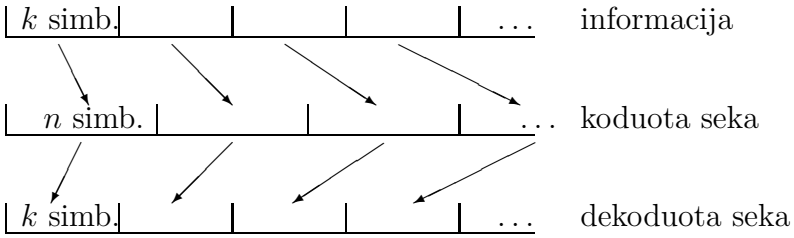
4 Blokinių ir konvoliucinių kodų pristatymas ir palyginimas

Klaidas taisančius kodus galima suskirstyti į dvi grupes: blokinius ir konvoliucinius (sąsūkos).

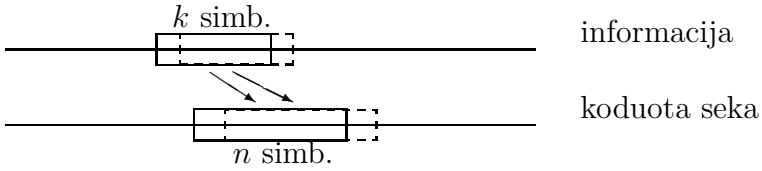
4.1 Blokiniai kodai

Informaciją skaidome į k ilgio blokus ir kiekvieną jų, nepriklausomai vieną nuo kito, užkoduojame n ilgio blokais:



4.2 Konvoliuciniai kodai

Turime informacijos srautą, kuriame bet kuriuo laiko momentu užkoduota informacija priklauso nuo k paskutinių informacijos simbolių:



4.3 Palyginimas

Konvoliucinių kodų privalumai, lyginant su blokiniais:

1. Greitas kodavimas ir dekodavimas, vykstantys vienu metu su informacijos siuntimu ir gavimu;
2. Nekyla problemų dėl sinchronizacijos (išskyrus pačią perdavimo pradžią);
3. Kodavimo ir dekodavimo schemos labai paprastos, tuo tarpu blokiniam kodui sunku rasti efektyvius dekodavimo algoritmus.

Konvoliucinių kodų trūkumai, lyginant su blokiniais:

1. Mažesnės klaidų ištaisymo galimybės nei blokiniame kodavime;
2. Jų struktūra mažiau ištyrinėta, tuo tarpu blokinių kodų struktūra yra įvairesnė ir geriau išstudijuota, nes blokiniai kodai yra artimesni tradicinėms, gerai ištyrinėtoms matematinėms struktūroms.

4.4 Konvoliucinio kodo pavyzdys

4.4.1 Kodavimas

Tarkime turime begalinę dvinarę matricą G:

$$G = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

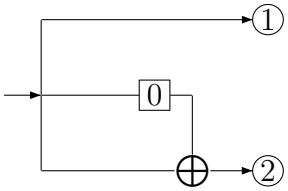
Begalinį informacijos vektorių m užkoduojame daugindami iš matricos G. Pavyzdžiui, jei

$$m = (1\,0\,1\,1\,0\,0\ldots),$$

tai jis užkoduojamas vektoriumi

$$c = m \cdot G = (1\,1\,0\,1\,1\,1\,1\,0\,0\,1\,0\,0\ldots).$$

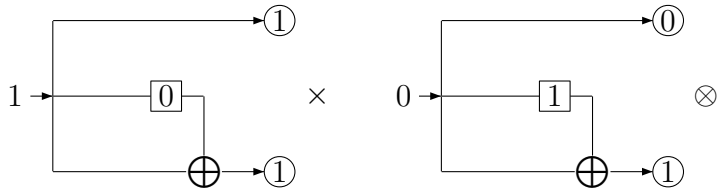
Tokį kodavimą paprasčiau galime realizuoti, naudodami *stumiamuosius registrus* (angl. *shift register*). Mūsų pavyzdžio atveju gauti c galime naudodami tokį registrą:

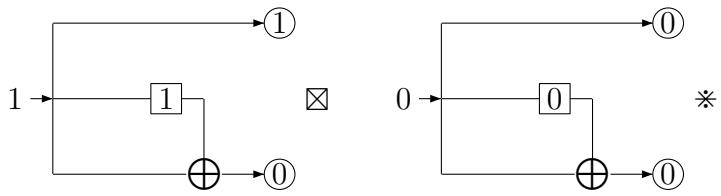


Čia stačiakampis yra atmintis, užlaikanti įrašytą reikšmę vieną laiko momentą. Pradiniu laiko momentu jos reikšmė yra nulis.

Į šią schemą rodyklės kryptimi įėjus vienam simboliui, išeina du. Pirmasis iš jų yra lygus įėjusiam (matome, kad kas tik įeina, tuoj viršutine atšaka ir išeina). Antrasis yra suma (moduliu 2) įėjusio simbolio ir iš atminties išėjusio simbolio, t.y. simbolio, įėjusio prieš tai. Taigi, antrasis yra paskutinio ir priešpaskutinio įėjusių simbolių suma. Be to, ėjęs simbolis nukeliauja ir į atmintį (vidurinė atšaka), iš kurios išeis kitu laiko momentu (įėjus kitam simboliui).

Panagrinėkime, kaip iš m gauname c naudodami duotą schemą. Brėžiniuose parodoma, kokie simboliai išeina, priklausomai nuo to, kas įėjo ir kas buvo atminty. Matom, kad pradžioje įeina 1, atminty yra 0, todėl išeina 11. Tada įeina 0, atminty — 1, išeina 01. Toliau kartojasi pirma situacija: įeina 1, atminty — 0, išeina 11. Tada įeina 1, atminty 1, išeina 10. Tuomet vėl kartojasi: įeina 0, atminty 1, išeina 01. Tada įeina 0, atminty 0, išeina 00. Ir t.t.





Gauname:

$$c = (\underbrace{1\,1}_{\times}\ \underbrace{0\,1}_{\otimes}\ \underbrace{1\,1}_{\times}\ \underbrace{1\,0}_{\boxtimes}\ \underbrace{0\,1}_{\otimes}\ \underbrace{0\,0}_{\divideontimes}\ldots)$$

Matome, kad naudodami stumiamąjį registrą, gavome lygiai tokį patį vektorių c, kaip ir daugindami iš matricos G. Kodėl taip yra?

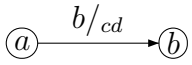
Matricos G nelyginiuose stulpeliuose yra vienetiniai (turintys vienetą vienoje iš pozicijų ir nulius visose kitose) vektoriai, o lyginiuose stulpeliuose yra lygiai po du vienetus (išskyrus antrą stulpelį), vadinasi jei $m = (m_1, m_2, m_3, \ldots)$, tai vektorius c, gautas sudauginus m su G, bus toks:
$\begin{cases} c_{2k-1} = m_k; \\ c_{2k} = m_{k-1} + m_k; \end{cases}$, kur $k \geqslant 1$, o $m_0 = 0$.

Kaip matėme, duotas stumiamasis registras konstruoja būtent tokį vektorių c.

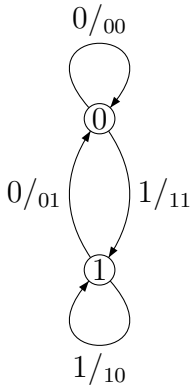
4.4.2 Būsenų diagrama

Galima nubrėžti konvoliucinio kodo *būsenų diagramą*. Konvoliucinio kodo *būsena* vadinsime jo stumiamojo registro atminties turinį.

Tarkime, kodo būsena yra a, į schemą įeina koks nors simbolis b, išeina simboliai c ir d, būsena keičiasi į b. Tai žymėsime tokiu būdu:



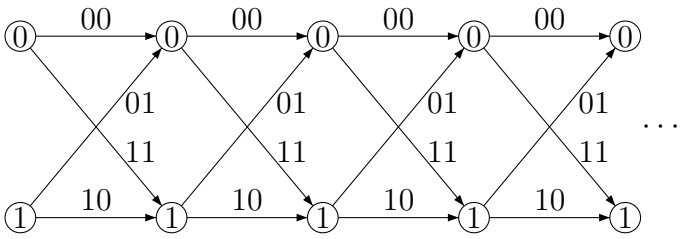
Tada duoto kodo būsenų diagrama būtų tokia:



Šita diagrama leidžia lengvai koduoti. Užkoduota seka atitinka kelią šiame grafe.

4.4.3 Dekodavimas

Būsenų diagramą išplečiame laike. Gauname:



Jei persiunčiant nepadaroma klaidų — lengvai grafe galima atsekti kelią ir dekoduoti žodį, pradedant nuo nulinės būsenos ir einant briaunomis, kurios atitinka iš kanalo gautą vektorių c'.

O kaip ištaisyti padarytas klaidas?

Iš anksto pasirenkame $m \geqslant 1$, ir imame m blokų iš eilės, lyginame su galimais keliais ir renkamės tą, kuris mažiausiai skiriasi.

Imkime pavyzdį. Tegu

$$c = (\underbrace{1\,1}_{1}\ \underbrace{0\,1}_{0}\ \underbrace{1\,1}_{1}\ \underbrace{1\,0}_{1}\ \underbrace{0\,1}_{0}\ldots)$$

Tarkime, klaida padaryta antrojoje vektoriaus pozicijoje, t.y.

$$c' = (1\,\underline{0}\,0\,1\,1\,1\,1\,0\,0\,1\ldots)$$

Tegu $m = 2$. Imkime pirmus m blokų, t.y. vektorių - $u = (1\ 0\ 0\ 1)$, ir randame atstumą (t.y. besiskiriančių koordinačių skaičių) iki kiekvieno galimo vektoriaus, gauto iš būsenų diagramos (išplėstos laike). Iš būsenų diagramos matome, kad galimi 4 vektoriai (einame iš nulinės būsenos ir žiūrime, kokie vektoriai galėjo būti gauti koduojant):

$$\begin{array}{ll} (0\,0\,0\,0) & 2 \\ (0\,0\,1\,1) & 2 \\ (1\,1\,0\,1) & 1 \\ (1\,1\,1\,0) & 3 \end{array}$$

Čia prie kiekvieno vektoriaus nurodytas jo atstumas iki u. Mažiausias atstumas yra trečiojo iš galimų vektorių, todėl juo keičiame u.

Tada dekoduojam pirmąjį bloką, t.y. 11. Dekodavę gauname 1.

Dabar kartojame viską su nauju u. Vėl imame m blokų: $u = 0111$. Būsenų diagramoje esame antro laiko momento būsenoje 1. Iš ten irgi yra keturi keliai, iš kurių renkamės artimiausią (šiuo atveju 0111) ir dekoduojame antrąjį bloką 01 į 0. Ir t.t.

Taigi, vieną klaidą ištaisė.

Dabar tarkime, kad yra dvi klaidos — antroje ir ketvirtoje vektoriaus pozicijose, t.y.

$$c' = (1\,\underline{0}\,0\,\underline{0}\,1\,1\,1\,0\,0\,1\ldots)$$

Imame pirmąjį vektorių $u = (1\ 0\ 0\ 0)$ ir randame atstumą iki kiekvieno galimo vektoriaus. Pagal būsenų diagramą galimi vektoriai bei atstumai nuo u iki kiekvieno iš jų yra:

$$\begin{array}{ll} (0\ 0\ 0\ 0) & 1 \\ (0\ 0\ 1\ 1) & 3 \\ (1\ 1\ 0\ 1) & 2 \\ (1\ 1\ 1\ 0) & 2 \end{array}$$

Mažiausias atstumas yra iki pirmojo iš galimų vektorių, todėl juo keičiame pirmąjį vektorių. Gautą pirmąjį bloką 00 dekoduojame 0. Toliau imame antrąjį vektorių $u = (0\ 0\ 1\ 1)$, jis yra tarp galimų vektorių, todėl antrojo vektoriaus nekeičiame, antrą bloką 00 dekoduojame 0. Taip pat nekeičiame ir tolimesnių vektorių, dekuoduojame:

$$c = (\underbrace{1\ 0}_{\underline{0}}\ \underbrace{0\ 0}_{0}\ \underbrace{1\ 1}_{1}\ \underbrace{1\ 0}_{1}\ \underbrace{0\ 1}_{0}\ldots)$$

Dekodavome klaidingai. Taigi, dviejų klaidų jau nebeištaisė.

Toliau nagrinėsime tik blokinius kodus.

5 Bendrosios sąvokos

5.1 Kodas ir kodavimas

5.1 apibrėžimas. Abėcėle *vadinsime baigtinę netuščią aibę.*

Tarkime, turime abėcėlę A. Jos elementų skaičių paprastai žymėsime q. Jei $q = 2$, tai paprastai laikysime, kad $A = \{0, 1\}$, ir tokią abėcėlę vadinsime *dvinare.*

Tarkime, $n \geqslant 1$ yra sveikasis skaičius. Kaip įprasta, A^n žymėsime aibę visų n ilgio vektorių, kurių koordinatės priklauso aibei A:

$$A^n = \{(z_1, \ldots, z_n) \mid z_i \in A\ \forall i\}.$$

5.2 apibrėžimas. *Aibės A^n poaibį C vadinsime* blokiniu kodu virš A *arba tiesiog* kodu. *Vektorius, priklausančius kodui, vadinsime* kodo žodžiais. *Parametrą n vadinsime* kodo ilgiu*, o kodo žodžių skaičių M* — kodo dydžiu. *Tokį kodą žymėsime (n, M). Jei $q = 2$, kodą vadinsime* dvinariu.

Pavyzdys. $A = \{0, 1\},\ n = 5,\ C = \{(0, 0, 0, 0, 0), (1, 1, 0, 0, 0), (0, 1, 1, 1, 1)\},\ M = 3.$ □

Tarkime, C yra (n, M) kodas. Tegu $k \geqslant 1$ yra toks sveikasis skaičius, kad $q^k \leqslant M$ (kadangi $M \leqslant q^n$, tai $k \leqslant n$). Laikome, kad informacija, kurią norime persiųsti nepatikimu ryšio kanalu, yra abėcėlės A simbolių seka. Ją skaidome į k ilgio informacijos vektorius m. Tarkime, turime kokią nors injekciją $c : A^k \to C$ (tokia injekcija egzistuoja, nes $q^k \leqslant M$). Naudodami šią injekciją, užkoduosime informacijos vektorius m, tiksliau, vektorių $m \in A^k$ užkoduosime vektoriumi $x = c(m) \in C$. Paprastai laikysime, kad c yra bijekcija, t.y. $c(A^k) = C$ (tam turi būti $q^k = M$). Tokiu atveju egzistuos atvirkštinė funkcija $c^{-1} : C \to A^k$, kurią galėsime naudoti dekodavimo metu.

Pavyzdys. Tarkime, $A = \{0, 1\}$, ir turime ilgio $n = 3$ kodą $C = \{000, 101, 011, 110\} \subset A^3$. Tegu $k = 2$, tada $q^k = M = 4$. Apibrėžkime bijekciją $c : A^k \to C$ taip: $c(00) = 000$, $c(01) = 101$, $c(10) = 011$ ir $c(11) = 110$. Tada informacijos srautą skaidome į $k = 2$ ilgio vektorius, ir kiekvieną galimą vektorių koduojame, naudodami bijekciją c, pavyzdžiui, informacijos vektorių $m = 01$ koduojame vektoriumi $x = c(01) = 101$. Dekodavimui naudosime atvirkštinę funkciją $c^{-1} : C \to A^k$, kuri yra tokia: $c^{-1}(000) = 00$, $c^{-1}(101) = 01$, $c^{-1}(011) = 10$, $c^{-1}(110) = 11$. □

Matome, kad kodo dydis M turi būti kuo didesnis, kad juo būtų galima užkoduoti kuo daugiau informacijos vektorių (t.y. kuo M didesnis, tuo didesnį k, tenkinantį $q^k \leqslant M$, galime parinkti). Tačiau, kuo daugiau žodžių kode, tuo mažiau jie skiriasi vienas nuo kito ir tuo sunkiau nustatyti, kuris vektorius buvo pasiųstas, ir surasti klaidos pozicijas.

Reikia rinktis tokį kodą, kurio ir dydis, ir atstumai tarp kodo žodžių yra kuo didesni. Tai prieštaraujančios viena kitai sąlygos. Parenkant kodus, ieškoma kompromiso tarp šių dviejų dalykų.

5.2 Atstumas ir svoris

5.3 apibrėžimas. *Jei* $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in A^n$, *tai* Hemingo atstumas *tarp vektorių* x *ir* y, *žymimas* $d(x, y)$, *yra koordinačių, kuriose jie skiriasi, skaičius:*

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

Įsitikinkite patys, kad Hemingo atstumas yra *metrika*, t.y. $\forall x, y, z \in A^n$ tenkina šias savybes:

1. $d(x, y) \geqslant 0$. Be to, $d(x, y) = 0$ tada ir tik tada, kai $x = y$.
2. $d(x, y) = d(y, x)$.
3. $d(x, y) \leqslant d(x, z) + d(z, y)$ (trikampio nelygybė).

5.4 apibrėžimas. *Tarkime, kad abėcėlė* A *yra Abelio grupė sudėties atžvilgiu. Vektoriaus* $x \in A^n$ Hemingo svoris*, žymimas* $w(x)$*, yra jo nenulinių koordinačių skaičius:*

$$w(x) = |\{i : \ 1 \leqslant i \leqslant n, x_i \neq 0\}|.$$

Nuo šiol laikysime, kad mūsų naudojama abėcėlė A yra Abelio grupė sudėties atžvilgiu.

Nesunku įsitikinti, kad:

1. $w(x) = d(x, 0)$.
2. $d(x, y) = w(x - y) = w(y - x)$.
3. $w(x + y) \leqslant w(x) + w(y)$ (trikampio nelygybės analogas).

Įrodykite šias savybes.

Kadangi šiame kurse naudosime tik Hemingo atstumą ir svorį, tai juos paprastai vadinsime tiesiog *atstumu* ir *svoriu*.

5.3 Dekodavimo taisyklės

Taigi, visą informaciją, kurią norime išsiųsti, skaidome į k ilgio vektorius $m \in A^k$, kiekvieną vektorių užkoduojame n ilgio kodo žodžiu $x = c(m) \in C$ ir siunčiame į kanalą. Iš kanalo gauname iškraipytą vektorių $y \in A^n$, kuris gali nebepriklausyti kodui C, t.y. y yra bet kuris vektorius iš aibės $\mathcal{A}^n$. Dekodavimo metu vektoriui y priskiriamas vektorius $m' \in \mathcal{A}^k$. Dekodavimas paprastai vykdomas dviem etapais. Visų pirma vektoriui y priskiriame kodo žodį $x' \in C$, t.y. ištaisome kanale padarytas klaidas. Antrame etape tiesiog pasinaudojame bijekcijos c atvirkštine funkcija $c^{-1} : C \to \mathcal{A}^k$, kad žodžiui $x' \in C$ priskirtume $m' \in \mathcal{A}^k$. Pirmame dekodavimo etape naudojama funkcija $f : \mathcal{A}^n \to C$ vadinama *dekodavimo taisykle*.

Pavyzdys. Prisiminkime pakartojimo kodą R_3. Dekodavimas yra 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, dekoduojame imdami artimiausią (matuojant Hemingo atstumu) kodo žodį (pirmas etapas), ir tada imdami informacijos vektorių, atitinkantį tą kodo žodį (antras etapas). 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. □

Tarkime, ryšio kanalu gavome vektorių $y \in A^n$. Koks kodo žodis $x \in C$ buvo išsiųstas? Norėtųsi rasti tokį kodo žodį x, kurio išsiuntimo tikimybė yra didžiausia, žinant, kad gavome vektorių y. Dekodavimo taisyklė, kai iš kanalo gautą vektorių y dekoduojame kodo žodžiu $x \in C$, maksimizuojančiu tikimybę $P(x|y)$ (taip žymėsime sąlyginę tikimybę, kad į kanalą buvo išsiųstas x, jei žinome, kad iš kanalo gautas y), vadinama *idealaus stebėtojo taisykle*. T.y., dekoduojame tokiu kodo žodžiu x, kad $P(x|y) = \max_{x' \in C} P(x'|y)$.

Be abejo, šią dekodavimo taisyklę sunku naudoti praktikoje, nes tas tikimybes skaičiuoti yra sudėtinga. Nesunkiai galima parodyti, kad esant išpildytoms tam tikroms sąlygoms, ši dekodavimo taisyklė sutampa su *minimalaus atstumo dekodavimo taisykle*: dekoduojame tuo kodo žodžiu, kuris yra artimiausias gautam vektoriui y, matuojant Hemingo atstumu, t.y. gavę $y \in A^n$, jį dekoduojame tokiu kodo žodžiu $x \in C$, kad $d(x, y) = \min_{x' \in C} d(x', y)$.

5.5 teorema. *Jei naudojame q-narį simetrinį kanalą su simbolių iškraipymo tikimybe $p < \frac{q-1}{q}$, ir jei visi kodo C žodžiai į kanalą perduodami su vienodomis tikimybėmis, tai idealaus stebėtojo ir minimalaus atstumo dekodavimo taisyklės duoda tą patį rezultatą.*

Įrodymas. Tarkime, $x = (x_1, \ldots, x_n) \in C,\ y = (y_1, \ldots, y_n) \in A^n,\ u = d(x, y)$.

Teoremą įrodysime, jei parodysime, kad u yra minimalus tada ir tik tada, kai sąlyginė tikimybė $P(x|y)$ - maksimali.

Pagal sąlyginių tikimybių savybes $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. Kadangi visi kodo C žodžiai į kanalą perduodami su vienodomis tikimybėmis, tai $P(x) = \frac{1}{M}$, kur $P(x)$ — kodo žodžio x išsiuntimo į kanalą tikimybė, o M — kodo C dydis. Taigi, $P(x|y) = \frac{P(y|x)}{MP(y)}$. Kadangi y — fiksuotas, tai

$$P(x|y) \text{ — maksimali} \quad \Longleftrightarrow \quad P(y|x) \text{ — maksimali.}$$

Naudojamas kanalas yra be atminties, todėl kiekvieno išsiųsto simbolio gavimo tikimybė yra nepriklausoma nuo kitų simbolių tikimybių. Kadangi $u = d(x, y)$, tai buvo padaryta lygiai u klaidų. Todėl

$$P(y|x) = \prod_{i=1}^{n} P(y_i|x_i) = (1-p)^{n-u}\left(\frac{p}{q-1}\right)^u = (1-p)^n\left(\frac{p}{(1-p)(q-1)}\right)^u.$$

Bet $p < \frac{q-1}{q}$, todėl $\frac{p}{(1-p)(q-1)} < 1$. Iš tikro,

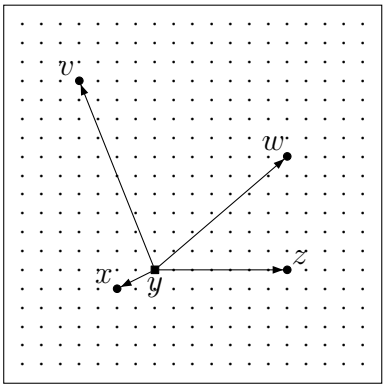
$$\frac{p}{(1-p)(q-1)} < 1 \Leftrightarrow p < (1-p)(q-1) = q - 1 - pq + p \Leftrightarrow pq < q - 1 \Leftrightarrow p < \frac{q-1}{q}.$$

Taigi, kuo u mažesnis, tuo $P(y|x)$ yra didesnė, nes $\frac{p}{(1-p)(q-1)} < 1$. Ką ir reikėjo įrodyti. □

Pastaba. $p < \frac{q-1}{q} \Leftrightarrow 1 - p > \frac{p}{q-1}$ - t.y. teoremos sąlygos reikalavimas ekvivalentus reikalavimui, kad neklaidos tikimybė būtų didesnė už bet kokios klaidos tikimybę.

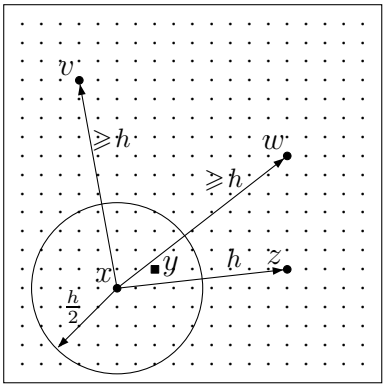
5.4 Dekodavimas ir minimalus atstumas

Dekodavimą naudojant minimalaus atstumo dekodavimo taisyklę grafiškai galima pavaizduoti taip:



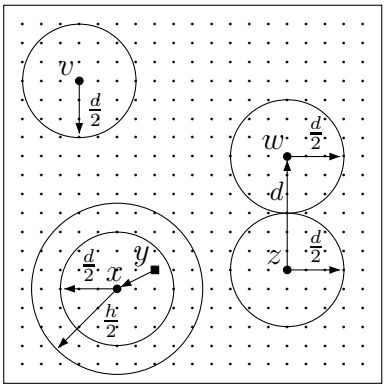
Šiame pavyzdyje kvadratas su taškais yra aibė visų vektorių iš $\mathcal{A}^n$, didesni taškai priklauso kodui $C = \{v, w, x, z\} \subseteq \mathcal{A}^n$. Iš kanalo gautas vektorius y (žymimas kvadratuku) gali būti bet kuris taškas. Pažiūrime, kuris kodo žodis yra arčiausiai, juo ir dekoduojame (čia rodyklės rodo tai, kad matuojame atstumus tarp y ir kiekvieno kodo žodžio). Šiuo atveju arčiausiai yra kodo žodis x, todėl juo ir dekoduojame.

Matome, kad kodo žodžiui x artimiausias kitas kodo C žodis yra z. Pažymėkime h atstumą tarp x ir z. Tada, jei iš kanalo gautas vektorius y yra nutolęs nuo x mažesniu atstumu, nei $h/2$, tai jis tikrai bus dekoduotas žodžiu x, nes šis kodo C žodis bus arčiausias. Taigi, apie kodo žodį x galime apibrėžti spindulio $h/2$ apskritimą, ir visi jo viduje esantys taškai bus dekoduojami to apskritimo centru x:



Tuo pačiu tai reiškia, kad jei į kanalą buvo pasiųstas kodo žodis x, ir kanale buvo padaryta mažiau kaip $h/2$ klaidų, iš kanalo išėjęs žodis y tikrai bus dekoduotas teisingai (visos klaidos bus ištaisytos), nes y bus apskritimo, apibrėžto apie x, viduje. Jei klaidų padaryta $h/2$ arba daugiau, dėl teisingo dekodavimo jau nesame garantuoti — kai kada gali dekoduoti teisingai, kai kada klaidingai. Taigi, turime tarsi kodo „gerumo“ („kokybiškumo“) matą, parodantį, kiek kodas tikrai ištaisys klaidų, jei į kanalą buvo pasiųstas kodo žodis x.

Tą patį galime padaryti su kiekvienu kodo žodžiu, ir gausime daugybę įvairaus spindulio apskritimų, apibrėžtų apie kodo žodžius. Bet norėtųsi turėti universalų kodo „gerumo“ („kokybiškumo“) matą, nepriklausantį nuo kodo žodžių. Tam iš visų apskritimų spindulių išrenkame mažiausią, tarkim, tai yra $d/2$, apie kiekvieną kodo žodį apibrėžiame spindulio $d/2$ apskritimą, ir tada esame garantuoti, kad jei kanale buvo padaryta mažiau kaip $d/2$ klaidų, iš kanalo išėjęs žodis y tikrai bus dekoduotas teisingai (nepriklausomai nuo to, koks kodo žodis buvo pasiųstas į kanalą), nes jis bus teisingo apskritimo viduje:



Toks d vadinamas kodo minimaliu atstumu. Pateikiame formalų jo apibrėžimą.

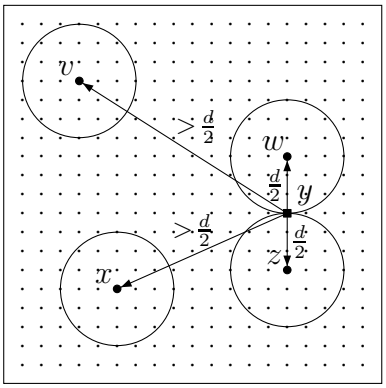
5.6 apibrėžimas. *Tarkime, C yra (n, M) kodas ir $M \geqslant 2$ (kodas C sudarytas iš bent dviejų žodžių). Kodo C* minimalus atstumas *d yra mažiausias Hemingo atstumas tarp dviejų skirtingų kodo C žodžių, t.y. $d = \min_{x,y\in C,\, x\neq y} d(x, y)$. (n, M) kodą, kurio minimalus atstumas yra d, žymėsime (n, M, d).*

Tarkime, $M \geqslant 1$. Kodo C minimalus svoris *w yra mažiausias nenulinio kodo C žodžio svoris, t.y. $w = \min_{x\in C,\, x\neq 0} w(x)$.*

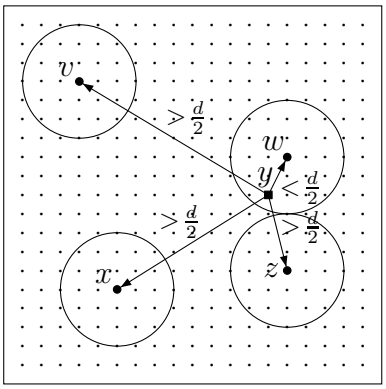
5.7 pavyzdys. Tarkime, $\mathcal{A} = \{0, 1\}$, $C = \{00001, 11101, 00110\}$. Atstumai tarp kodo žodžių yra tokie: $d(00001, 11101) = 3$, $d(00001, 00110) = 3$ ir $d(11101, 00110) = 4$. Mažiausias atstumas yra trys, todėl kodo C minimalus atstumas $d = 3$. Taigi, kodas C yra $(5, 3, 3)$ kodas.

Kodo C nenulinių žodžių svoriai yra tokie: $w(00001) = 1$, $w(11101) = 4$ ir $w(00110) = 2$. Todėl kodo C minimalus svoris yra 1. □

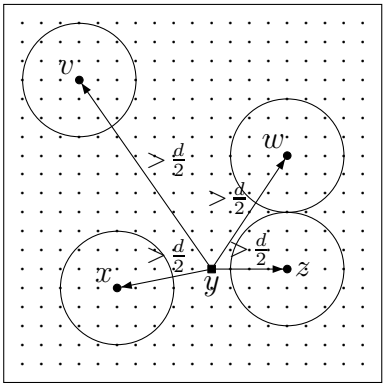
Tarkime, į kanalą pasiuntėme žodį x, ir kanale jame buvo padaryta t klaidų. Iš kanalo išėjo vektorius y, esantis atstumu t nuo x. Jei $t < d/2$, tai y bus kodo žodžio x apskritimo viduje, todėl dekoduosime žodžiu x, t.y. ištaisysime kanalo padarytas klaidas. Jei $t = d/2$, tai y gali priklausyti dviem apskritimams, ir tokiu atveju nežinosime, kurio apskritimo centru dekoduoti. Tokia situacija, kai į kanalą buvo pasiųstas kodo žodis z, kanale buvo padaryta $d/2$ klaidų, ir iš kanalo išėjęs vektorius y yra vienodai nutolęs nuo kodo žodžių z ir w, yra pavaizduota šiame brėžinyje:



Jei $t > d/2$, tai y gali atsidurti jau kito kodo žodžio skritulyje, ir tokiu atveju bus dekoduotas neteisingai. Pavyzdžiui, šiame brėžinyje pavaizduota, kad, į kanalą pasiuntus kodo žodį z, kanale buvo padaryta daugiau nei $d/2$ klaidų, ir iš kanalo išėjęs vektorius pateko jau į kito kodo žodžio w apskritimo vidų:



Jei iš kanalo išėjęs vektorius y neatsiduria jokio kodo žodžio skritulyje, tai, jį dekoduodami artimiausiu kodo žodžiu, galime dekoduoti teisingai, o galime ir klaidingai. Pavyzdžiui, šiame brėžinyje pavaizduota, kad, į kanalą pasiuntus kodo žodį z, kanale buvo padaryta daugiau nei $d/2$ klaidų, bet arčiausias iš kanalo išėjusiam žodžiui y visgi išlieka z, ir dekoduota bus teisingai:



Taigi, jei klaidų skaičius $t \geqslant d/2$, nesame garantuoti dėl dekodavimo teisingumo.

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.*

Taigi, kodas C bus tiksliai t klaidų taisantis kodas, jei t yra didžiausias toks skaičius, kad kodas C yra t klaidų taisantis kodas.

5.8 pavyzdys. 5.7 pavyzdžio kodas C yra tiksliai 1 klaidą taisantis kodas. Iš tikrųjų, nesunku patikrinti, kad kanale įvykus vienai klaidai, artimiausias vistiek liks siųstas kodo žodis, todėl, naudojant minimalaus atstumo dekodavimo taisyklę, dekoduojama bus teisingai (patikrinkite patys!). O įvykus dviems klaidoms, dekoduoti gali ir klaidingai. Pavyzdžiui, jei į kanalą buvo pasiųstas kodo žodis $c = 00001$, kanale įvyko dvi klaidos, iš kanalo išėjo $y = 11001$, tai artimesnis vektoriui y bus kitas kodo žodis $c' = 11101$, todėl, naudojant minimalaus atstumo dekodavimo taisyklę, dekoduojama bus klaidingai. □

Teorema. *Kodas C yra tiksliai $[(d-1)/2]$ klaidų taisantis kodas, kur d yra kodo C minimalus atstumas, o $[a]$ yra skaičiaus a sveikoji dalis, t.y. didžiausias sveikas skaičius, mažesnis arba lygus už a.*

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 sveikas skaičius, mažesnis už $d/2$. Jei d yra lyginis, tai didžiausias sveikas skaičius, mažesnis už $d/2$, yra $d/2 - 1 = (d-2)/2 = [(d-1)/2]$. Jei d yra nelyginis, tai didžiausias sveikas 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.

5.9 pavyzdys. 5.7 pavyzdžio kodo C minimalus atstumas $d = 3$, todėl pagal teoremą jis yra tiksliai $[(d-1)/2] = 1$ klaidą taisantis kodas. □

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:

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.*

5.10 pavyzdys. 5.7 pavyzdžio kodas C yra tiksliai 2 klaidas aptinkantis kodas. Iš tikrųjų, nesunku patikrinti, kad kanale įvykus vienai ar dviem klaidoms, gautas vektorius nepriklausys kodui C. O įvykus trims klaidoms, iš kodo žodžio $c = 00001$ galime gauti kitą kodo žodį $c' = 11101$, todėl kodas C nėra tris klaidas 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:

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

5.11 pavyzdys. 5.7 pavyzdžio kodo C minimalus atstumas $d = 3$, todėl pagal teoremą jis yra tiksliai $d - 1 = 2$ klaidas aptinkantis kodas. □

Taigi, jei norime sukonstruoti gerą kodą, tai stengiamės, kad ir jo dydis, ir minimalus atstumas būtų kuo didesni. Tai vienas kitam prieštaraujančios sąlygos, todėl ieškoma kompromiso.

6 Ekvivalentūs kodai

Kai užkoduojame pranešimą ir siunčiame kodo žodį kanalu be atminties, tai nėra skirtumo, kuria tvarka persiunčiame to kodo žodžio simbolius. Taigi, šiuo požiūriu kodai, kurie skiriasi tik simbolių eilės tvarka, iš esmės nesiskiria. Tokius kodus vadinsime *ekvivalenčiais*.

Kad galėtume duoti formalų ekvivalenčių kodų apibrėžimą, prisiminkime keitinio (perstatos) sąvoką.

Tarkime, turime baigtinę aibę $I = \{1, \ldots, n\}$. Abipusiškai vienareikšmį atvaizdį (bijekciją) $\sigma : I \to I$ vadina *perstata* (arba *keitiniu*). Pavyzdžiui, jei $n = 5$, tai $\sigma : I \to I$, apibrėžta taip: $\sigma(1) = 2$, $\sigma(2) = 3$, $\sigma(3) = 4$, $\sigma(4) = 5$, $\sigma(5) = 1$, yra perstata. Perstata dažnai užrašoma tokiu pavidalu:

$$\sigma = \begin{pmatrix} 1 & \ldots & n \\ \sigma(1) & \ldots & \sigma(n) \end{pmatrix}.$$

Pavyzdžiui, paskutinio pavyzdžio perstata galėtų būti užrašyta taip:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}.$$

Perstatas galime naudoti vektorių koordinačių keitimui vietomis. Laikysime, kad $I = \{1, \ldots, n\}$ — vektoriaus $x = (x_1, \ldots, x_n)$ indeksų aibė. Tada sukeičiame vektoriaus x koordinates taip, kaip nurodo perstata σ, t.y. pirmą koordinatę x_1 perkeliame į $\sigma(1)$ vietą, antrą — į $\sigma(2)$ vietą ir t.t. Gautą vektorių žymėsime $\sigma(x)$. Pavyzdžiui, jei $x = (x_1, x_2, x_3, x_4, x_5)$, tai panaudoję paskutinio pavyzdžio perstatą σ, gausime vektorių $\sigma(x) = (x_5, x_1, x_2, x_3, x_4)$. Iš tiesų, pirmą koordinatę x_1 perkeliame į antrą poziciją ir t.t., o pirmoje vietoje atsiduria x_5, nes $\sigma(5) = 1$. Nesunku pastebėti, kad jei $x = (x_1, \ldots, x_n)$, tai

$$\sigma(x) = (x_{\sigma^{-1}(1)}, \ldots, x_{\sigma^{-1}(n)}), \tag{2}$$

kur σ^{-1} — atvirkštinė σ perstata, t.y. tokia perstata, kad $\sigma^{-1}(\sigma(i)) = i$ $\forall i$. Pavyzdžiui, paskutinio pavyzdžio perstatos σ atvirkštinė perstata yra tokia perstata:

$$\sigma^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 2 & 3 & 4 \end{pmatrix}.$$

Taigi, perstata $\sigma : I \to I$ apibrėžia funkciją $\sigma : A^n \to A^n$, užduotą (2) lygybe. Ši funkcija tiesiog sukeičia vektoriaus koordinates vietomis.

Jei C yra kodas, žymėsime $\sigma(C) = \{\sigma(x) : x \in C\}$, t.y. $\sigma(C)$ bus kodas, gautas sukeitus visų kodo C žodžių koordinates pagal tą pačią perstatą σ.

6.1 apibrėžimas. *Du ilgio n kodai C ir C' vadinami* ekvivalenčiais*, jei egzistuoja tokia indeksų aibės $\{1, \ldots, n\}$ perstata σ, kad $\sigma(C) = C'$.*

Nesunku parodyti, kad ekvivalenčių kodų ilgiai, dydžiai, minimalūs atstumai, žodžių svorių pasiskirstymai sutampa (nes sukeitus vektoriaus koordinates vietomis, vektoriaus ilgis bei svoris išlieka tokie patys).

6.2 pavyzdys. Nustatykime, ar dvinariai kodai

$$C = \{1010, 1101, 0011, 1001\} \quad \text{ir} \quad C' = \{1010, 1100, 0110, 1011\}$$

yra ekvivalentūs, ir jei yra, raskime tokią perstatą σ, kad $C' = \sigma(C)$.

Pastebėsime, kad šį uždavinį galima išspręsti išsamiosios paieškos metodu, patikrinant visas galimas perstatas. Bet ilgio n perstatų yra $n!$, todėl jei n yra didelis, šis metodas nėra efektyvus. Tačiau yra sukurta įvairių taisyklių, padedančių sumažinti galimų perstatų skaičių iki priimtino. Pasinaudosime paprasčiausiomis iš jų.

Visų pirma panagrinėkime abiejų kodų žodžius. Matome, kad abiejų kodų žodžių svorių pasiskirstymai sutampa: kodai turi po tris svorio 2 žodžius ir po vieną svorio 3 žodį. Jei nesutaptų, iškart galėtume tvirtinti, kad kodai neekvivalentūs. Kadangi turi tik po vieną svorio 3 žodį, tai aišku, jog jei tokia perstata σ egzistuoja, tai jinai žodį 1101 atvaizduoja į 1011. Iš čia iškart gaunam, kad jei σ egzistuoja, tai $\sigma(3) = 2$, nes nulis iš trečios žodžio 1101 pozicijos perkeliamas į antrą žodžio 1011 poziciją. Taigi, gavome šiek tiek informacijos apie σ.

Toliau galime suskaičiuoti, kiek vienetų stovi pirmo kodo žodžių kiekvienoje pozicijoje, ir palyginti su antro kodo vienetų skaičiumi. Kad aiškiau matytųsi, užrašykime kodo žodžius stulpeliu:

$$C = \left\{ \begin{array}{c} 1010 \\ 1101 \\ 0011 \\ 1001 \end{array} \right\} \quad \text{ir} \quad C' = \left\{ \begin{array}{c} 1010 \\ 1100 \\ 0110 \\ 1011 \end{array} \right\}.$$

Perstatant kodo žodžių koordinates, šie stulpeliai yra perkeliami į kitą vietą, taigi, vienetų skaičius juose išlieka toks pat. Kodo C pirmame stulpelyje yra 3 vienetai, antrame — 1, trečiame — 2, ketvirtame — 3. Kodo C' atitinkamai 3,2,3,1. Vėlgi, matome, pasiskirstymas sutampa — jei nesutaptų, kodai nebūtų ekvivalentūs. Matome, kad antras kodo C stulpelis, kuriame yra 1 vienetas, būtinai keliauja į ketvirtą poziciją (t.y. jei kodai ekvivalentūs ir perstata σ egzistuoja, tai $\sigma(2) = 4$), ir trečias — į antrą (tą mes jau matėme, palyginę kodo žodžių svorius).

Taigi, jau žinome dvi perstatos σ reikšmes: $\sigma(2) = 4$ ir $\sigma(3) = 2$. Yra tik dvi perstatos su tokiomis reikšmėmis:

$$\sigma' = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix} \quad \text{ir} \quad \sigma'' = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 2 & 1 \end{pmatrix}$$

Patikriname jas ir gauname, kad jos abi tinka, t.y. abi jos kodo C koordinates sukeičia taip, kad gauname kodą C'. Iš tikro,

$$\sigma'(C) = \{1100, 1011, 0110, 1010\} = C' \quad \text{ir} \quad \sigma''(C) = \{0110, 1011, 1100, 1010\} = C'.$$

Taigi, kodai ekvivalentūs. □