Uroš Lotrič

Kodi

Kodiranje je pomembno pri stiskanju podatkov, popravljanju napak pri prenosu in šifriranju. Algoritmi za stiskanje podatkov poskušajo podatke predstaviti tako, da ohranijo vsebino in pri tem uporabijo občutno manj prostora. Pri prenosu podatkov se napakam zaradi motenj ne moremo izogniti, na srečo pa se lahko pred njimi dobro zavarujemo. S šifriranjem podatkov želimo preprečiti, da jih na poti od pošiljatelja do prejemnika ne bi prestregli vsiljivci in se z njimi okoristili.

V večini današnjih računalnikov so podatki predstavljeni z nizi, sestavljenimi iz dveh simbolov. Simbola sta lahko leva in desna stran, bela in črna kroglica, ničla in enica. Na primer, slovenska abeceda ima 25 različnih črk. Ena od možnosti je, da vsako črko predstavimo z nizom dolžine pet, v katerem nastopata samo dva različna simbola. Iz nizov dolžine pet lahko zgradimo 2^5=32 različnih vzorcev, kar je dovolj za 25 črk slovenske abecede in še za nekaj posebnih znakov. Lahko bi si zamislili, da znaku A priredimo niz 00001, znaku B niz 00010, … Takemu postopku prirejanja nizov pravimo kodiranje, preslikavi pa kod. Kod preslika vsak znak osnovne abecede v kodno zamenjavo, ki je sestavljena iz znakov kodne abecede.

Nekateri kodi so zelo razširjeni. Eden bolj uporabljanih kodov v računalništvu je kod ASCII, ki na podoben način kot zgoraj osnovnim znakom prireja nize dolžine sedem:

1 → 00110001, 2 → 00110010, …
A → 01000001, B → 01000010, …
a → 11000001, b → 11000010

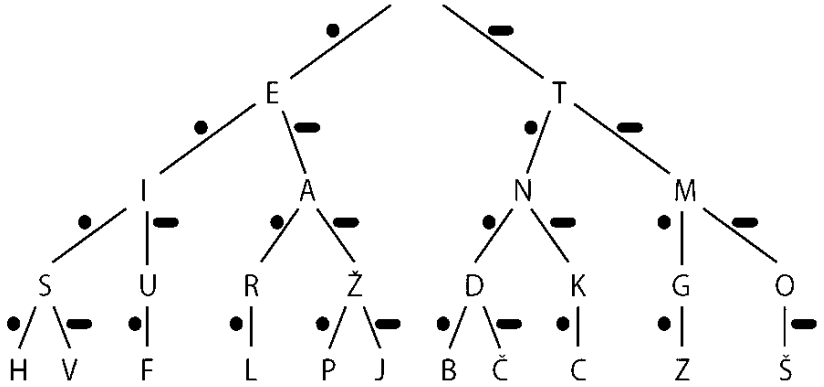
Kodi so seveda lahko tudi precej drugačni. Znaki so lahko kodirani tudi z različno dolgimi nizi ali kodnimi zamenjavami:

A → 0, B → 0 1, C → 0 1 0 .

Za potrebe telegrafiranja so sredi 19. stoletja razvili Morsejev kod. Sestavljen je iz kratkih in dolgih piskov ter pavz. Na papir se te znake zapisuje kot piko, črto in presledek. Najbolj pogosta črka angleške abecede E ima za kodno zamenjavo piko, malenkost manj pogosta črka A pa piko in črto. Kod za vsako črko angleške abecede in za vsako cifro uporablja edinstveno kodno zamenjavo.

Vsakodnevno se srečujemo še s črtno kodo, ki je ravno tako kod, pri katerem kodno abecedo sestavljata tanka in debela črta.

Kod je lahko kakršna koli preslikava. Najlepše ga predstavimo s kodnim drevesom. Vozlišča v drevesu označujejo znake, pot od korena do izbranega vozlišča pa nam določi kodno zamenjavo. Na primer, pri sprehodu po Morsejevem kodnem drevesu od korena do znaka C sestavimo kodno zamenjavo črta, pika, črta, pika.



Seveda vse preslikave niso smiselne in med smiselnimi tudi niso vse enako dobre. Kod ni singularen, če ima vsak znak svojo kodno zamenjavo. Pri takem kodu lahko iz kodnih zamenjav rekonstruiramo osnovne znake. Pri singularnem kodu to ne gre. Primer:

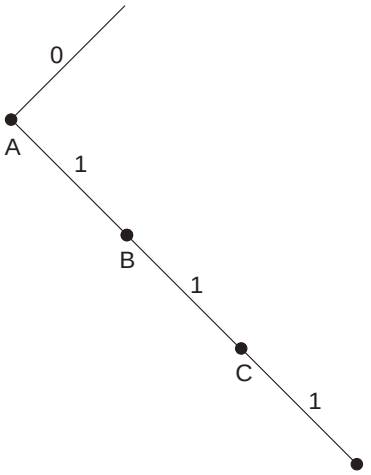
Znak	Singularni kod	Nesingularni kod
A	00	0
B	10	10
C	01	00
D	10	01

Zgornji nesingualrni kod še vedno ni najboljši, saj kodni zapis 00 lahko razumemo kot AA ali pa kot C. Podobno težavo srečamo tudi pri Morsejev kodu: ena črta predstavlja T, dve črti predstavljata M. Morsejev kod se je v zgodovini vseeno veliko uporabljal. Kako so se izognili težavi?

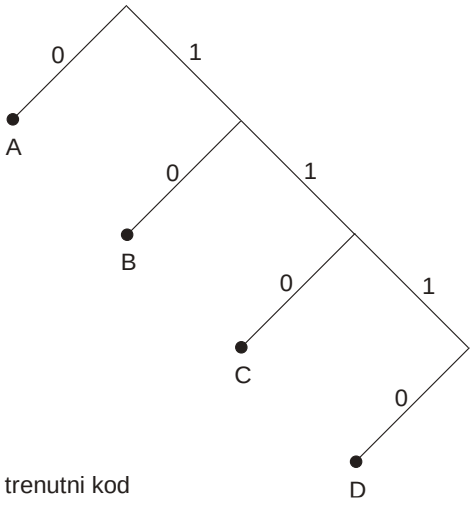
Uporabni so enoznačni kodi, za katere velja, da jih lahko rekonstruiramo na en sam način. Zaradi čim hitrejše obdelave podatkov se največ uporabljajo trenutni kodi. To so kodi, pri katerih lahko osnovni znak rekonstruiramo takoj ko smo prejeli zadnji znak kodne zamenjave. Spodnja tabela kaže dva podobna koda.

Znak	Enoznačni	Trenutni
A	0	0
B	01	10
C	011	110
D	0111	1110

Prvi je samo enoznačni, saj znak lahko rekonstruiramo iz števila zaporednih enic šele potem, ko prejmemo prvi znak (0) kodne zamenjave naslednjega znaka. Veliko boljši je trenutni kod, saj je 0 vedno zadnji znak kodne zamenjave – ko jo opazimo, preštejemo enke pred njo in rekonstruiramo znak. Naslednja slika prikazuje enoznačni in trenutni kod iz zgornje tabele. Trenutni kodi se od ostalih razlikujejo po tem, da imajo kodne zamenjave samo na listih, ne pa tudi na vmesnih vozliščih.



enoznačni kod



trenutni kod

Naloge, v katerih se skrivajo osnovne ideje kodiranja, vključujejo zamenjevanja znakov osnovne abecede v znake kodirne abecede, enoznačnost kodov in dopolnjevanje kodne preslikave.

053

Kodiranje

Bobra Barbara in Benjamin sta si izmislila nenavaden način zapisovanja besedil s številkami. Sestavila sta tabelo na desni. Vsaki črki ustreza številka; ko želita zapisati določeno črko, zapišeta dvakratnik ustrezne številke. Črko B zapišeta s številko 4; črko M zapišeta z 38. Besedo BOBER bi zapisala kot 45241256.

A	1	B	2	C	3	Č	4	D	5
E	6	F	7	G	8	H	9	I	15
J	16	K	17	L	18	M	19	N	25
O	26	P	27	R	28	S	29	Š	35
T	36	U	37	V	38	Z	39	Ž	45

Katero besedo predstavlja številka 562874502503212?

Številke stanovanj

Bober Peter opazuje sosednji kompleks mravljišč. Sestavljen je iz desetih mravljišč. Vsako ima pet vhodov. Za vsakim vhodom je 12 nadstropij. Prvo nadstropje ima 120 prostorov, ostala pa 80.

Mravlje so prostore oštevilčile tako, da prva številka označuje številko mravljišča (1-10), druga številko vhoda, tretja nadstropje in četrta prostor v tem nadstropju. Tako ima prostor 2 nadstropja 3 vhoda 4 mravljišča 8 številko 8432.

Petru se zdi to oštevilčenje nespametno. Pravi, da se bo pogosto dogajalo, da bodo imeli različni prostori isto številko. In res, od spodnjih štirih številk je kar tri možno brati na različne načine. Le za eno jasno, kateri prostor predstavlja. Za katero?

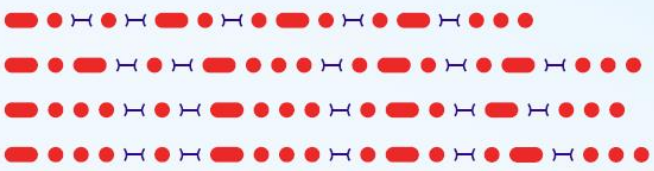
11122 14111 12131 11113

Piskajoči bobri

Pasma piskajočih bobrov si pošilja sporočila s kratkimi in dolgimi piski (zapisovali jih bomo s pikami in vejicami). Tule je nekaj črk:

Med črkami delajo kratke presledke, ki jih bomo zapisali z znakom

Eno od spodnjih zaporedij predstavlja besedo BEBRAS. Katero?



Stiskanje podatkov

Stiskanje je postopek, ki zmanjša velikost datotek, ne da bi resno zmanjšali integriteto podatkov. Stiskanje je na veliko uporabljano v moderni informacijski tehnologiji za zmanjšane tekstovnih, glasbenih, slikovnih datotek in video vsebin. Obstaja množica tehnik za stiskanje.

V grobem ločimo stiskanje brez izgub in stiskanje z izgubami. Medtem ko je pri prvem pomembno, da lahko stisnjene podatke raztegnemo nazaj v prvotno obliko, pri drugem dovolimo, da nekaj vsebine izgubimo. Slednji postopki se uporabljajo veliko pri stiskanju slik ter zvočnih in video zapisov. Izkoriščajo dejstvo, da ljudje zaradi fizične omejitve čutil ne zaznamo vseh podrobnosti, ki jih zabeležijo tipala v digitalnih napravah.

Vzemimo, da govorimo abecedo, ki pozna presledek in tri črke {_, A, B, C}. Kod, ki bi ga lahko uporabili je _ - 00, A – 01, B – 10, C – 11. Daljši niz v naši abecedi bi potem na primer kodirali takole:

A_CAA_ABA_BCAA_A → 01001101010001100100101101010001

Pa znamo bolje?

Huffmanov kod

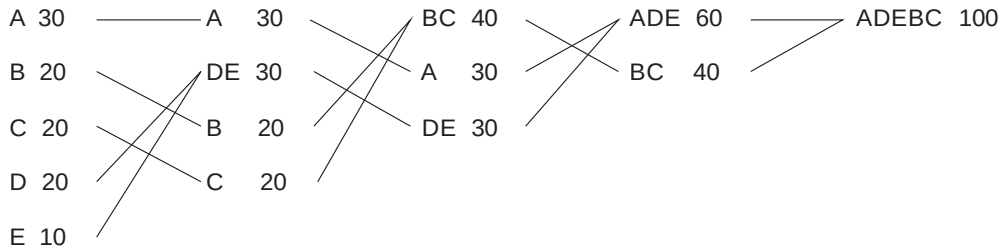
Osnovna ideja pri stiskanju je, da bolj pogoste znake zapišemo s krajšimi kodnimi zamenjavami, manj pogoste pa z daljšimi kodnimi zamenjavami. To idejo najbolje udejanja Huffmanov kod. V nadaljevanju si bomo ogledali, kako se z njim dobi kodne zamenjave, ki uporabljajo binarno kodno abecedo.

Vzemimo preprost primer, ko želimo kodirati dva znaka {A, B}. Prvemu bomo dodelili kodno zamenjavo 0, drugemu kodno zamenjavo 1.

Bolj zapleteno je, če želimo kodirati tri znake {A, B, C}. Predpostavimo, da je znak A najbolj pogost znak v našem zapisu. Huffman je ugotovil, da je najbolj smiselno najmanj pogosta znaka (B in C) združiti v nov znak BC. Zdaj, ko imamo samo dva znaka, jima lahko dodelimo kodni zamenjavi tako kot v prejšnjem primeru: A nadomestimo z 0, BC pa z 1. Sestavljeni znak BC razdružimo tako, da k enici, dodeljeni sestavljenemu znaku, enkrat pripišemo ničlo, drugič pa enico. Dobimo končne kodne zamenjave: A: 0, B: 10, C: 11. Huffmanov kod ni enolično določen – enako dobro bi bilo, če bi znakom dodelili naslednje kodne zamenjave: A: 0, B: 11, C: 10 ali pa A: 1, B: 01, C: 00.

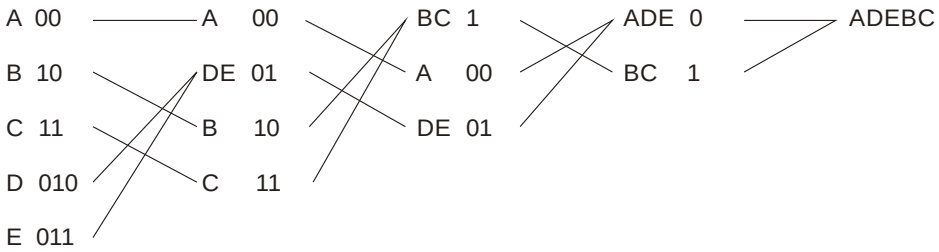
Posplošimo. Huffmanov postopek kodiranja poteka v dveh fazah. V prvi fazi pare najmanj pogostih znakov združujemo v sestavljene znake. Postopek ponavljamo, dokler nam ne ostane en sam sestavljeni znak. V drugi fazi sestavljene znake razdružujemo, pri čemer enemu (sestavljenemu) znaku dodelimo 0, drugemu pa 1.

Poglejmo primer. Imamo pet znakov {A, B, C, D, E}. Število posameznih znakov v zapisu je {30, 20, 20, 20, 10}. Postopek kodiranja gradnje Huffmanovega koda je grafično prikazan na spodnji sliki.

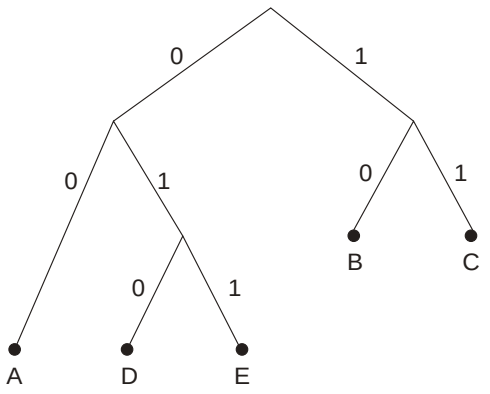


Najprej vzamemo najmanj pogosta znaka, to sta gotovo E in katerikoli od znakov B, C, D. Če izberemo D, nam ostanejo štirje znaki {A, DE, B, C } s pogostostmi {30, 30, 20, 20}. Šteli smo, da se znak DE pojavi, če se pojavi znak D ali znak E. Postopek nadaljujemo na enak način. Spet združimo najmanj pogosta znaka, to sta B in C. Dobimo znake {BC, A, DE} in pogostosti pojavitve {40, 30, 30}. Postopek ponovimo še dvakrat. Najprej dobimo {ADE, BC}, pri čemer se znaki A, D ali E pojavijo 60-krat, znaka B ali C pa 40-krat. Na koncu oba znaka združimo še v en sam znak.

V drugi fazi vsak sestavljen znak razdelimo na znaka, iz katerih smo ga v prvi fazi sestavili. Enemu od teh dveh znakov dodelimo znak kodne abecede 0, drugemu pa 1. Vzemimo, da znaku na zgornji veji dodelimo 0, znaku na spodnji veji pa 1. Dodelitev kodnih zamenjav je prikazana na spodnji sliki.



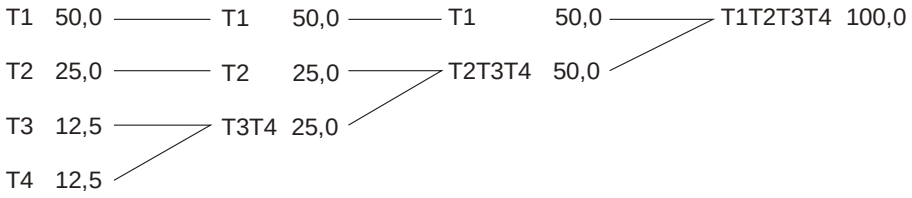
Znak ADEBC sestavljata znaka ADE in BC. Znak ADE je narisan zgoraj, zato mu dodelimo 0, znaku BC pa 1. V naslednjem koraku se znak BC ne razcepi, zato se njegova kodna zamenjava ne spremeni. Znak ADE se razcepi na A in DE, zato 0 iz prejšnjega koraka pri A dopišemo 0, pri DE pa 1. Če na enak način nadaljujemo do konca, dobimo kodne zamenjave, ki jih bolj elegantno predstavimo s kodnim drevesom.

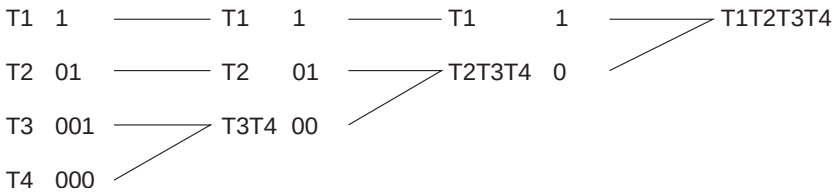


Najbolj verjetni znak A kodiramo samo z dvema znakoma kodne abecede, najmanj verjetni znak E pa s tremi.

Kodne zamenjave si lahko predstavljamo tudi kot odgovore na vprašanja, ki nas od korena pripeljejo do želene črke. Vprašanja morajo biti zastavljena tako, da na njih lahko odgovorimo z ne (0) ali z da (1).

Poglejmo primer. Na tekmi so nastopili štirje tekmovalci {T1, T2, T3, T4}. Verjetnost za zmago prvega je bila 50 %, drugega 25 %, tretjega in četrtega pa 12,5 %. Tekme nismo spremljali, radi pa bi s čim manj vprašanji izvedeli, kdo je zmagal. Zato najprej zgradimo Huffmanovo drevo. Ali lahko z verjetnostjo delamo tako, kot smo prej s pogostostjo pojavitve?

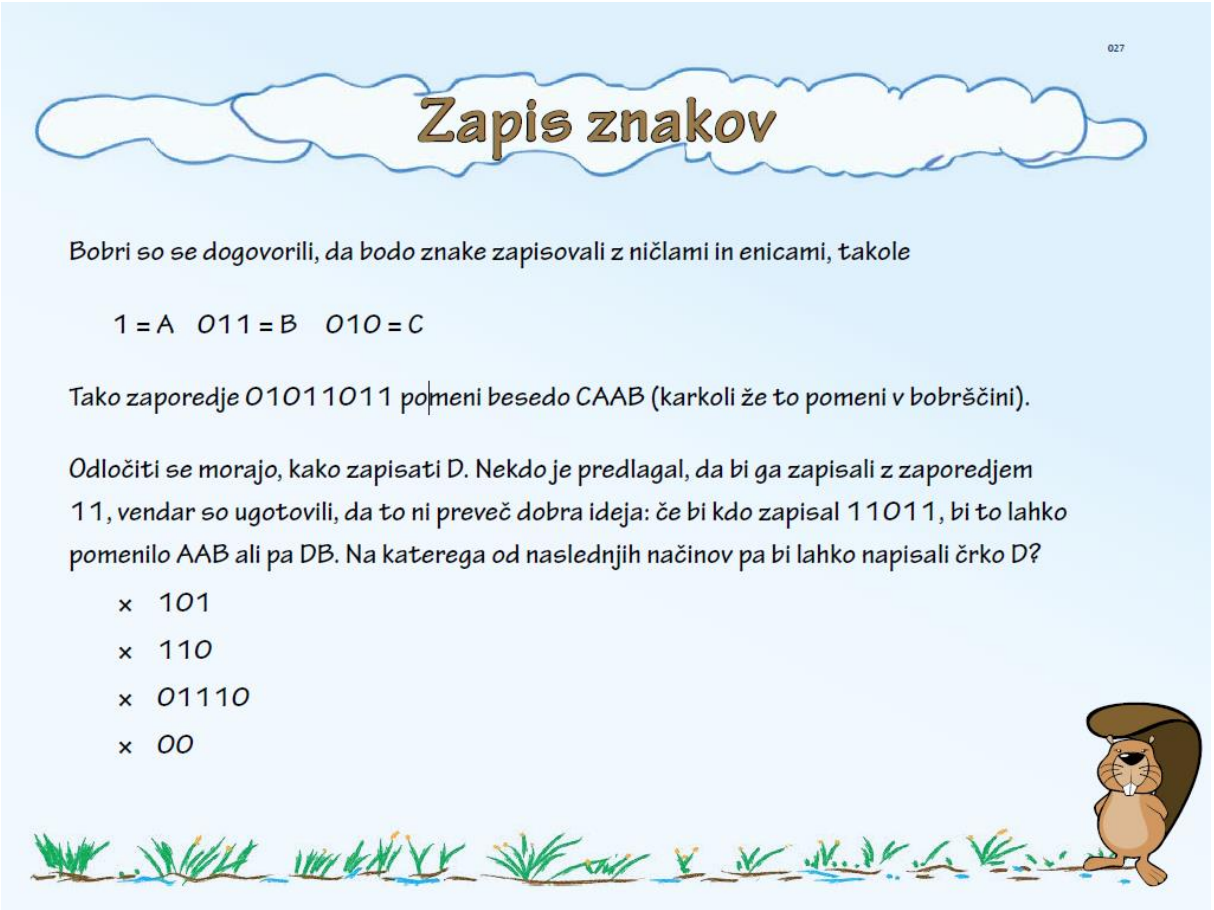




Na podlagi drevesa poskusimo oblikovati vprašanja. Tekmovalca T1 od ostalih ločimo z vprašanjem »Ali je zmagal tekmovalec T1?« To je tudi najbolj smiselno vprašanje, saj je zelo verjetno, da je zmagal. Drugo vprašanje bi bilo »Ali je zmagal tekmovalec T2?«, saj je verjetnost za njegovo zmago bistveno večja od verjetnosti za zmago tekmovalca T3 ali tekmovalca T4. Če ni zmagal niti tekmovalec T1 niti tekmovalec T2, z vprašanjem »Ali je zmagal tekmovalec T3?« dokončno razblinimo dvome.

Opisani postopek gradnje kodnih zamenjav uporablja večina modernih algoritmov za stiskanje podatkov (ZIP, RAR, JPG, …) . Največji problem pri stiskanju podatkov v praksi je, da ne poznamo verjetnosti. Različni postopki za stiskanje se tako razlikujejo predvsem v načinu ocenjevanja verjetnosti.

Poznavanje trenutnih kodov in risanje kodnih dreves se skriva za nalogama Zapis znakov in Bobrovsko kodiranje.



BOBROVSKO KODIRANJE

Bobri prevažajo hlode po reki s splavom. Hlodi so po velikosti lahko majhni (M), srednje velki (S), veliki (V) ali zelo veliki (Z). Da označijo, kakšni hlodi so na splavu, uporabljajo poseben kodirni sistem. Kadar splav pride do jezu, s pomočjo opozorilnega roga in posebnih kod opišejo hlode na splavu, recimo MZZZMVVS. Na opozorilnem rogu znajo zapiskati nizek (○) in visok ton (◊). Med vsakim piskom je sekundni odmor. Bober Brane si je zamislil štiri kodirne tabele za velikosti hlodov, ki so opisane spodaj. Žal je zgolj ena sprejemljiva. Katera?

A) M: ◊	S: ◊◊	V: ◊◊◊	Z: ◊◊◊◊
B) M: ◊	S: ○	V: ◊◊	Z: ○◊
C) M: ◊	S: ○◊	V: ○○◊	Z: ○○○
D) M: ○	S: ○◊	V: ○○	Z: ○◊○

S Huffmanovim kodom si lahko pomagamo tudi pri nalogi kot je Ugibanje števila.

111

Ugibanje števila

Bobri so radi v šoli, le med odmori jim je dolgčas. Zato se pogosto igrajo ugibanje števil: eden si zamisli število med 1 in 100, drugi ga poskuša ugibati. Ta, ki si je zamislil število, ob vsakem ugibanju pove, ali je iskano število večje ali manjše.

Bobrovka Hana vedno ugiba tako, da začne pri številu 50. Če je iskano število manjše, bo nadaljevala s 25, če večje s 75. Tako nadaljuje: v vsakem koraku pove število, ki je na sredi med najmanjšim in največjim kandidatom.

Kolikokrat, največ, mora ugibati, preden ugane?

- × 7
- × 16
- × 40
- × 50

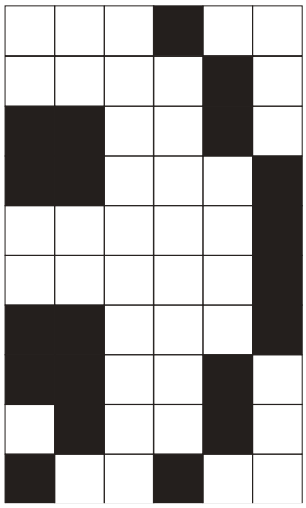
Verižni kodi

Kadar želimo kodirati znak za znakom, dobimo najboljše rešitve z gradnjo Huffmanovega koda. Če za to ni potrebe, lahko stiskamo še bolj temeljito. Ena od takih tehnik je verižno kodiranje.

Ideja verižnega kodiranja ali kodiranja z dolžinami nizov je zelo preprosta. Izkorišča dejstvo, da se v zapisih vzorci ponavljajo. Namesto večkratnega zapisovanja enakega vzorca verižno kodiranje zapiše en vzorec in število ponovitev. Na primer, niz znakov AAAABBC bi z verižnim kodiranjem zapisali kot 4A2B1C. Težava se pojavi, ko se znaki ne ponavljajo. Takrat z verižnem kodiranjem dobimo nasprotni učinek – zapis se podaljša. Na primer niz ABCBAC bi zapisali kot 1A1B1C1B1A1C in namesto šestih uporabili kar dvanajst znakov! Običajno se verižno kodiranje zato uporablja samo, kadar z njim skrajšamo zapis, drugače se kodira znak po znak. Niz znakov AAAABBC, verižno kodiran z omenjeno dopolnitvijo, je potem 4ABBC. Za enega ali dva enaka zaporedna znaka kodiranje z verižnim kodom ni smiselno, zato jih ohranimo v osnovnem zapisu.

Verižno kodiranje srečamo v napravah za faksiranje sporočil (standard ITU-T4), v slikovnih formatih BMP, TIFF, PCX.

Sliko, na primer, si lahko predstavljamo kot zaporedje točk, ki si sledijo vrstico za vrstico. Omejimo se na črno bele slike, na primer na tako, kot je spodnja.



Čisto osnovno verižno kodiranje nas pripelje do opisa

B3 Č1 B6 Č1 B1 Č2 B2 Č1 B1 Č2 B3 Č1 B5 Č1 B5 Č3 B3 Č3 B2 Č1 B2 Č1 B2 Č1 B1 Č1 B2 Č1 B2 .

Pri tem zapisu je prva težava ta, da moramo v naprej vedeti, kako široka je slika. Temu se lahko ognemo. Običajno kodiramo vsako vrstico posebej:

B3 Č1 B2 B4 Č1 B1 Č2 B2 Č1 B1 Č2 B3 Č1 B5 Č1 B5 Č1 Č2 B3 Č1 Č2 B2 Č1 B1
B1 Č1 B2 Č1 B1 Č1 B2 Č1 B2 .

V zgornjem zapisu so samo zaradi jasnosti med opise posameznih vrstic dodani večji presledki.

Za črno-bele slike lahko ta zapis v marsičem poenostavimo. Ni nam potrebno sporočati, koliko belih pik sledi zadnji črni, lahko samo povemo, da se je vrstica končala. Nadalje lahko upoštevamo, da se črna in bela barva izmenjujeta – beli sledi vedno črna in obratno. Edino, česat ne vemo, je, kakšne barve je prva točka v vrstici. Ena možnost je, da opis vrstice vedno začnemo z belo barvo, druga pa, da barvo vsakič pošljemo. Če uporabimo prvo idejo, bi bil zapis enak

3 1 | 4 1 | 0 2 2 1 | 0 2 3 1| 5 1 | 5 1 | 0 2 3 1 | 0 2 2 1 | 1 1 2 1 | 0 1 2 1 | ,

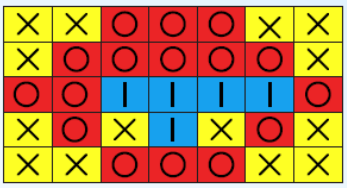
kjer navpičnica predstavlja znak za konec vrstice.

To idejo kodiranja pri pošiljanju podatkov uporabljajo faksi. V praksi je stvar še malo bolj zapletena. Za učinkovito pošiljanje so dolžine belih in črnih polj ter konce vrstic kodirali še z dvema Huffmanovima kodoma – kodom za bele in kodom za črne pike. Drevesa so zgradili po analizi pogostosti posameznih dolžin v množici tipičnih dokumentov.

Pri sestavljanju nalog Zapisovanje slike in Slika iz pik so bile uporabljene ravno zgornje ideje.

Zapisovanje slike

Kako bi spremenili sliko v zaporedje znakov? Eden od načinov je takšen:



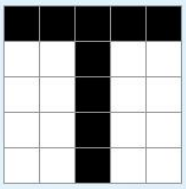
bxcobx
axeoax
???
axaoaxaiaxaoax
bxcobx

Vsako vrstico slike opisuje zaporedje na desni. Kako pa bi opisali manjkajočo srednjo vrstico?

Slika iz pik

Sliko, sestavljeno iz pobarvanih ali praznih kvadratkov na mreži, lahko opišemo s številkami, kot kaže slika.

0, 5
2, 1, 2
2, 1, 2
2, 1, 2
2, 1, 2



V vsaki vrstici se izmenjujejo beli in črni kvadratki. Prva številka pove, s koliko belimi kvadratki se začne slika, druga številka pove, koliko črnih jim sledi, tretja spet pove število belih kvadratkov, druga črnih in tako naprej, dokler je potrebno.

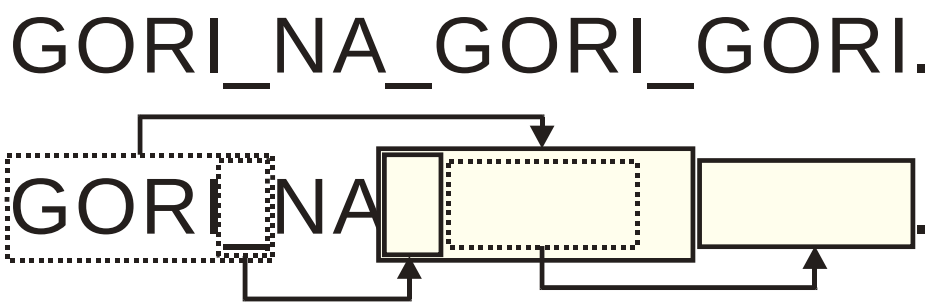
Katera črka bi se pokazala, če bi izrisali tole sliko?

0,1,3,1
0,1,3,1
0,5
0,1,3,1
0,1,3,1

Stiskanje z iskanjem podobnih vzorcev

Ta način stiskanja uporabljajo programi za stiskanje datotek, na primer ZIP. Ideja za to stiskanje prihaja iz analize jezikov, v katerih zaporedja črk niso popolnoma naključna, ampak se vzorci od časa do časa ponavljajo. Stiskanje je učinkovito, če si zapomnimo nize znakov, ki so se že pojavili. Ko najdemo enak niz znakov, nato elegantno sporočimo samo koliko znakov nazaj se je niz že pojavil in kako dolg je bil.

Poglejmo primer:



V tekstovni obliki lahko zapišemo v obliki

GORI_NA(3,1)(8,5)(5,4).

Pari številk v oklepajih predstavljajo sklic na obstoječi niz. Prva številka označuje, koliko znakov nazaj se niz začne, druga pa njegovo dolžino.

Obstoječi nizi se prepisujejo znak po znak, zato si lahko privoščimo tudi sklic na niz, ki je daljši od števila že zapisanih znakov:

BUM_(4,11)!

No, pravi algoritem LZ77 (Leta 1977 sta ga predlagala Lempel in Ziv), ki ga uporablja tudi program ZIP, vedno zapisuje trojčke (odmik, dolžina, naslednji znak). Zapis

GORI_NA_GORI_GORI.

pretvori v obliko

(0, 0, G)(0, 0, O)(0, 0, R)(0, 0, I)(0, 0, _)(0, 0, N)(0, 0, A)(3, 1, G)(8, 4, G)(5, 3, .)

Podobno koti pri faksiranju sporočil algoritem odmike in dolžine kodira s Huffmanovim algoritmom.

Algoritem LZW, imenovan po avtorjih Lempelu, Zivu in Welchu, namesto iskanja enakih vzorcev v nizu besedila sproti gradi slovar. Začne z osnovnim slovarjem in ga dopolnjuje z vsakim nizom, ki se pojavi v besedilu, v slovarju pa ga še ni. Poglejmo primer na besedilu

GORI_NA_GORI_GORI.

Vzemimo, da je osnovni slovar sestavljen samo iz znakov: A, G, I, N, O, R, _, . Vsakemu znaku priredimo številčno vrednost, na primer tako, kot je to prikazano v spodnji levi tabeli.

indeks	vpis
1	A
2	G
3	I
4	N
5	O
6	R
7	_
8	.

indeks	vpis
9	GO
10	OR
11	RI
12	I_
13	_N
14	NA
15	A_
16	_G
17	GOR
18	RI_
19	_GO
20	ORI
21	I.

Naša naloga je, da poiščemo najdaljši podniz, ki je že v slovarju. Poglejmo na našem primeru. Pri sprehodu čez besedilo opazimo, da imamo znak G že v slovarju, zato poskusimo še z nizom GO. Tega v slovarju ni, zato znaku G priredimo vrednost 2, niz GO pa dodamo v slovar pod naslednjo zaporedno številko (zgornja desna tabela). Iskanje nadaljujemo na enak način z znakom O. Ko se sprehodimo čez celotno besedilo, dobimo naslednji zapis:

G	O	R	I	_	N	A	_	G O	R I	_ G	O R	I	.
2	5	6	3	7	4	1	7	9	11	16	10	3	8

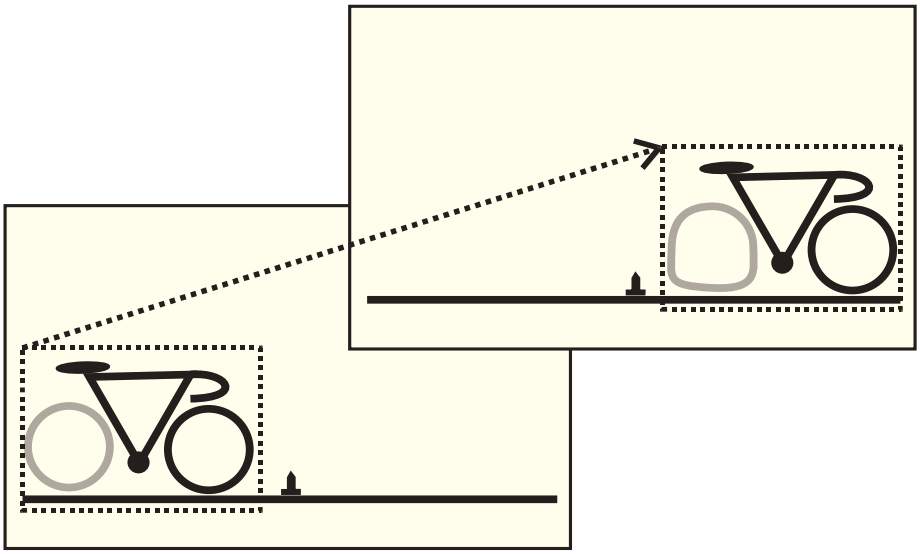
Dekodiranje tudi začnemo z osnovnim slovarjem, ki ga potem po vsakem sprejetem znaku dopolnimo. Ko sprejmemo 2, zapišemo G, ko sprejmemo 5, zapišemo O. Hkrati niz GO zapišemo v slovar. Potem sprejmemo 6, ki jo nadomestimo z R, v slovar pa vpišemo OR. Na ta način nadaljujemo do konca.

Huda zvijača: v slovarju imamo zapisani črki A in B z indeksoma 1 in 2. Kateri niz predstavljajo kode 1, 2, 3, 5?[1]

Stiskanje videa

Človeško oko lahko obdela deset do dvanajst slik na sekundo. Če ga izpostavimo hitrejšemu spreminjanju slik, ustvarimo navidezno gibanje. Na tej prevari sloni vsa filmska industrija. Danes se v filmih zamenja vsaj 24 slik vsako sekundo. Zaporedne slike so si zato med seboj zelo podobne. To lahko izkoristimo tudi pri stiskanju. Namesto, da bi stiskali vsako sliko posebej, lahko primerjamo dve zaporedni sliki. Enostavno samo povemo, v čem se nova slika razlikuje od prejšnje. Ta postopek izkoriščajo vsi moderni zapisi filmov (kodeki), na primer MPEG.

[1] ABABABA



Na zgornji sliki je dovolj, da povemo, za koliko se je spremenil položaj okvira, ki je orisan okrog kolesa in kako se je spremenila slika kolesa.

Nekaj podobnega počnejo tudi Bobri v nalogi Opisovanje filmov.

Varno kodiranje

Pri prenašanju in shranjevanju podatkov lahko pride do napak. Največkrat je vzrok v elektronskih vezjih, ki podležejo vplivom temperature ali zunanjih motenj. Pomislite na telefoniranje v mirnem in hrupnem okolju. Pogovor v hrupnem okolju bo potekal bistveno težje. Napake se lahko pojavijo kjerkoli in kadarkoli (prenosi podatkov, branje iz opraskane zgoščenke), vendar se je do neke mere mogoče obvarovati pred njimi ali jih celo popraviti. Ena možnost je, da izboljšamo zanesljivost fizičnega medija (tišje okolje pri telefoniranju), druga pa je uporaba shem za preverjanje pravilnosti podatkov, oziroma celo takih, ki znajo napake popravljati. Spet bomo gradili kode, le da tokrat poudarek ne bo na čim krajšem zapisu, temveč na povečevanju zanesljivosti v zameno za počasnejši prenos.

Če imamo v osnovni abecedi dva znaka, recimo A in B, in kodiramo binarno, enostavno priredimo enemu znaku eno vrednost, drugemu pa drugo, na primer 0 priredimo znaku A, 1 pa znaku B. Niz znakov AABBAB bi v tem primeru poslali kot 001101. Pri prenosu po žici ničlo največkrat predstavimo z nizko napetostjo, enico pa z visoko napetostjo. Zaradi zunanjih motenj, mogoče magnetnega polja, se lahko zgodi, da visoka napetost pade pod dovoljeno mejo. Na drugi strani bi potem namesto enice dobili ničlo. Če spet pogledamo naš primer – pri prenašanju tretjega znaka je prišlo do motenj in namesto 001101 smo na drugi strani sprejeli 000101. Rekonstrukcija nam prinese sporočilo AAABAB, ki je drugačno od poslanega.

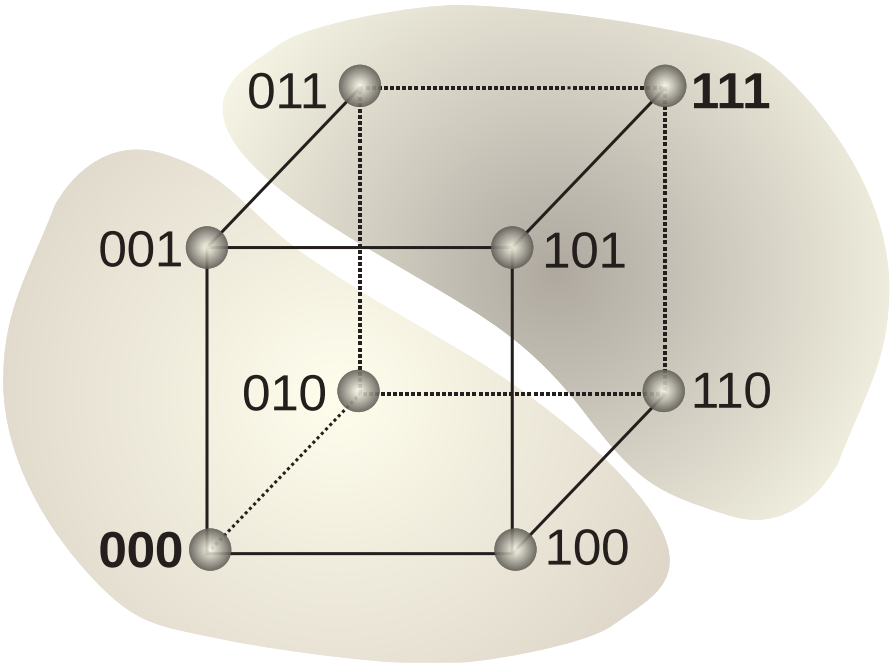
Ponavljajoče kode

Pomislimo, kako se pogovarjamo po telefonu v hrupnem okolju. Po potrebi prosimo sogovornika da ponovi izrečene besede. Na enak način lahko zagotovimo večjo jasnost tudi z računalnikom. Seveda je najbolj enostavno, da se računalniki ne dogovarjajo o ponovnem pošiljanju, ampak isti znak vsakič pošljejo večkrat. Namesto 0 in 1 lahko pošiljamo 00 in 11. Kodna zamenjava sedaj postane bistveno daljša. Za zgornji primer

AABBAB → 000011110011

vendar bistveno bolj zanesljiva. Kadar prejmemo 00 ali 11, vemo, da gre za znak A oziroma B, v primeru, da prejmemo 01 ali 10, pa vemo, da je prišlo pri prenosu do napake. Žal napake ne znamo popraviti. V zoprni situaciji, ko se narobe preneseta dva zaporedna znaka, pa naš sistem napake ne zazna. V primeru, da bi se zgornji niz prenesel kot 000000110011, bi narobe rekonstruirali osnovno sporočilo v AAABAB.

Več kot dodamo varnostnih znakov, bolje je. Pri uporabi treh binarnih simbolov za osnovni znak – A zamenjamo z 000 in B z 111 – je prenos bolj zanesljiv in seveda še počasnejši. Zdaj znamo napake celo popraviti. Če prejmemo več ničel kot enic, lahko sklepamo, da je bil poslan znak A, če je več enic kot ničel, pa znak B. Narišimo vse možne kombinacije v obliki grafa.



Edini pravi kodni zamenjavi sta odebeljeni. V okolici vsake od njiju pa so še tri kodne zamenjave, ki imajo vsaka po eno napako. Dobili smo dva otoka kodnih zamenjav, ki se med seboj nikjer ne prekrivata, zato lahko napake tudi popravljamo. Postopek pa seveda ni kos vsem napakam; če pride do dveh napak, recimo, da se 000 med prenosom spremeni v 101, pa znak ponovno narobe dekodiramo. Tu igramo na srečo, pri čemer se zavedamo, da so dvojne napake veliko manj verjetne od posameznih. Pa še posamezne napake naj bi se zgodile zelo malokrat.

Preverjanje sodosti

Na prej omenjeni način se dobro zavarujemo pred napakami, ni nam pa všeč, da prenos tako upočasnimo. V praksi se več uporabljajo kodi, ki na bolj zvite načine preverjajo skladnost podatkov. Velikokrat podatke razvrstimo v pravokotnik. Vzemimo niz znakov

ABBABABAABBABBA .

Znake osnovne abecede zamenjamo z ničlami in enicami (A → 0, B → 1) in jih zapišemo v obliki pravokotnika. Zapis predstavlja sivi pravokotnik na spodnji sliki

0	1	1	0	1
0	1	0	0	1
1	0	1	1	0

Za varnost lahko na koncu vsake vrstice in vsakega stolpca dodamo še en znak kodirne abecede tako, da je vsota v vsaki vrstici ali v vsakem stolpcu sodo število. Rezultat je v spodnji tabeli.

0	1	1	0	1	1
0	1	0	0	1	0
1	0	1	1	0	1
1	0	0	1	0	0

Ta koda lahko popravi eno napako, ne glede na katerem od 24 bitov se pojavi. Poglejmo nekaj primerov. Če je napaka v drugi vrstici in tretjem stolpcu, potem vsota v drugi vrstici in tretjem stolpcu ne bo dala sodega števila. Vrstica in stolpec z napako nam torej natančno določata bit, ki se je napačno prenesel. Nič drugače ni, če se je narobe prenesel kateri od varnostnih bitov.

Če se spremeni več bitov, vemo, da je nekaj narobe, ne znamo pa natančno ugotoviti kaj. Poglejmo, kaj se zgodi, če sta narobe prenesena prvi in drugi bit. Tabela je potem

1	0	1	0	1	1
0	1	0	0	1	0
1	0	1	1	0	1
1	0	0	1	0	0

Hitro vidimo, da je nekaj narobe v prvem in drugem stolpcu, ne vemo pa, v kateri vrstici. Napake ne moremo odpraviti.

Kaj pa kontrola lihosti? Ni problema, dokler je število obeh, stolpcev in vrstic, sodo ali liho število. Če temu ni tako, imamo težave s spodnjim desnim bitom.

V zgornjem primeru smo petnajst bitov, ki predstavljajo osnovni niz, varovali z devetimi varnostnimi biti. To je veliko bolje kot prej, ko smo vsak znak v osnovnem nizu varovali z dvema varnostnima bitoma. Prenosi so zato seveda hitrejši. Zato pa smo plačali tudi ceno. Prej smo znali popraviti eno napako v vsakem trojčku bitov, zdaj znamo zanesljivo popraviti eno napako v skupini 24 bitov.

Podobno shemo se da zasnovati tudi v obliki trikotnika. Spet vzemimo naš osnovni niz znakov ABBABABAABBABBA in ga z ničlami in enicami zapišimo v trikotnik

0	1	1	0	1
0	1	0	0	
1	1	0		
1	1			
0				

ter ga dopolnimo z biti za preverjanje sodosti. Zdaj bite nastavljamo tako, da je skupna vsota bitov v vrstici in stolpcu, v katerih je varnostni bit, soda:

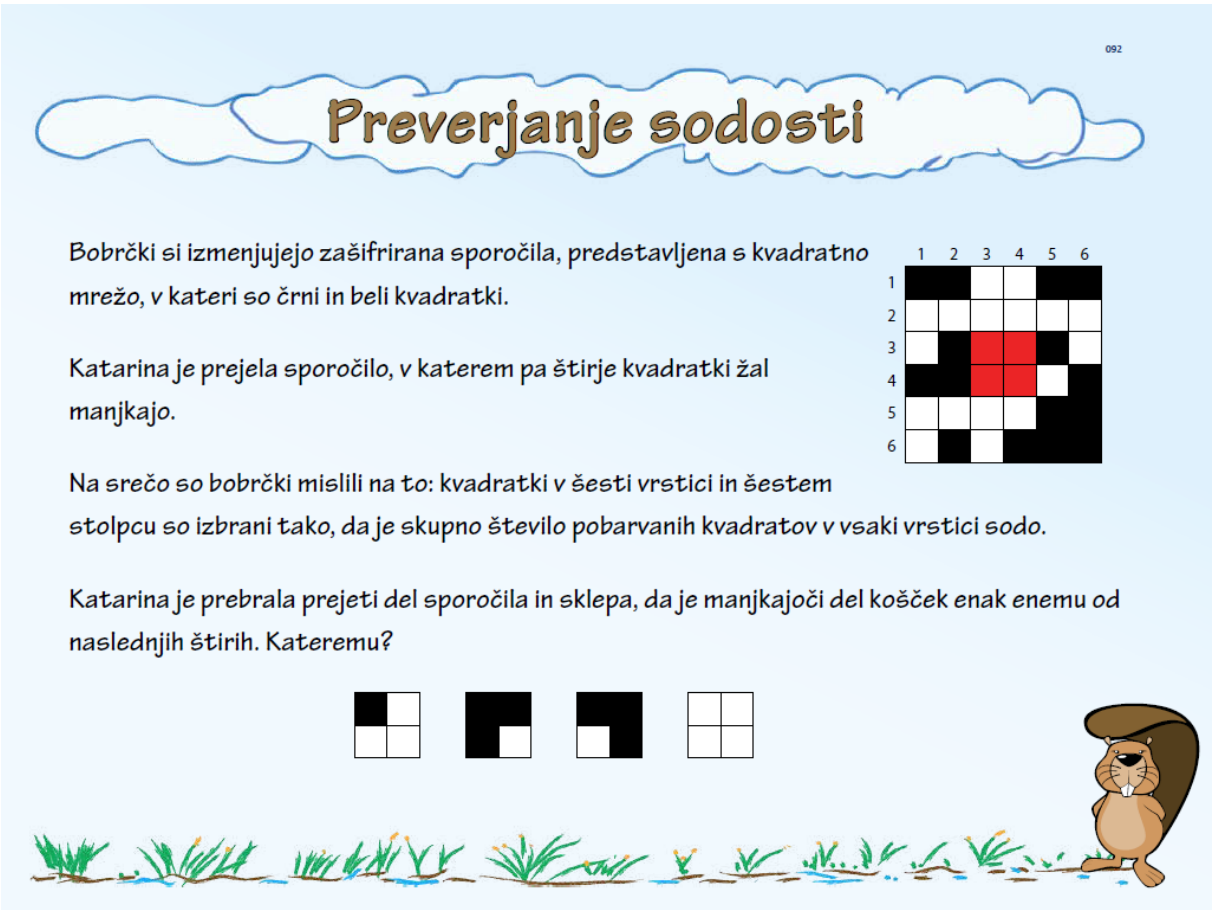
0	1	1	0	1	1
0	1	0	0	0	
1	1	0	0		
1	1	1			
0	0				
0					

Na zgornji sliki so biti, ki jih vključimo v računanje varnostnega bita v četrti vrstici in tretjem stolpcu, uokvirjeni. Tudi s tem kodom lahko odkrijemo in popravimo eno napako. Če je napaka na enem od bitov, ki predstavlja znak osnovnega niza, bosta napačna dva varnostna bita. Brez težav ugotovimo, za kateri bit gre. Če se narobe prenese varnostni bit, bo napaka ena sama. Vemo, da moramo obrniti ta bit.

Ta kod varuje petnajst bitov, ki predstavljajo osnovni niz, s šestimi varnostnimi biti. Pri popravljanju je enako zmogljiv kot pravokotnik kod, ker ima manj varnostnih bitov, zna odkriti manj dvojnih napak.

Z varnostnim kodiranjem ne moremo zagotoviti popolnoma varnega prenosa podatkov, lahko pa močno zmanjšamo možnost, da napake pri prenosu ne zaznamo. Kakšno varnostno kodiranje uporabiti, je močno odvisno od zahtev vsakega sistema posebej.

S preštevanjem kvadratkov se spopadajo tudi bobri v nalogah Preverjanje sodosti.



Varno kodiranje v vsakdanjem življenju

Po analogiji s preverjanjem parnosti lahko preverjamo tudi pravilnost zapisov številk, recimo bančnih računov, kod ISBN, črtnih kod in osebnih identifikacijskih številk.

Koda ISBN natančno določa knjigo. Koda ima od 10 do 13 števk. Poglejmo zapis z deset števkami:

0-691-12418-3 .

Prva števka določa jezik (0 za angleščino), naslednje dve ali tri pa določajo založnika (691 je Princeton University Press). Naslednjih pet števk določa oznako knjige, ki jo dodeli založnik. Zadnja številka je namenjena preverjanju pravilnosti kode in lahko zavzame vrednosti od 0 do 10. Če je zadnja vrednost 10, jo zamenjamo s črko X. Zadnji znak mora biti določen tako, da velja

$$z_1 + 2z_2 + 3z_3 + 4z_4 + 5z_5 + 6z_6 + 7z_7 + 8z_8 + 9z_9 + 10z_{10} = 0 \bmod 11 \quad .$$

Z z_1 do z_{10} so označeni znaki v kodi. Za naš primer lahko preverimo

$$0 + 2 \times 6 + 3 \times 9 + 4 \times 1 + 5 \times 1 + 6 \times 2 + 7 \times 4 + 8 \times 1 + 9 \times 8 + 10 \times 3 = 198 = 11 \times 18 = 0 \bmod 11$$

Koda je zasnovana tako, da zna odpraviti eno narobe zapisano števko in zamenjavo dveh zaporednih števk. Gre za dve najbolj pogosti napaki pri tipkanju.

V Sloveniji za identifikacijo oseb uporabljamo številko EMŠO. EMŠO je sestavljen iz 13 števk, recimo:

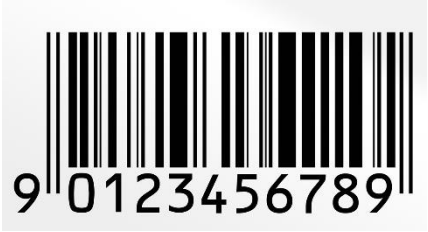
2809013505005 .

V zgornji kodi 28 označuje datum rojstva, 09 mesec rojstva, 013 leto rojstva brez tisočic, 50 Slovenijo (koda EMŠO je z nami še od časov Jugoslavije, druge republike so uporabljale druge številke območja), naslednje tri števke pa določajo spol in zaporedno številko rojstva na ta dan. Za moške zavzamejo vrednosti od 000 – 499, za ženske pa od 500 – 999. Zadnja števka je kontrolna. Določimo jo tako, da je vsota, ki jo dobimo po spodnjem receptu, deljiva z 11:

	2	8	0	9	0	1	3	5	0	5	0	0	5	
x	7	6	5	4	3	2	7	6	5	4	3	2	1	
	14	48	0	36	0	2	21	30	0	20	0	0	5	= 176

V našem primeru je ustrezna kontrolna številka 5, saj je 176/11 = 16. V primeru, da bi morali uporabiti kontrolno številko 10, zaporedno številko povečamo za 1 in ponovimo postopek.

Na izdelkih v trgovini najdemo črtno kodo. Obstaja množica različnih črtnih kod. Zaradi lažjega branja kod z elektronskimi bralniki je vsaka številka, napisana pod kodo, predstavljena z več debelimi ter tankimi črtami in presledki med njimi. Kadar je koda zmečkana ali pa slabo odtisnjena, elektronski bralnik zelo težko pravilno prebere vse števke. V takem primeru kontrolna vsota ne drži in prodajalka mora kodo vtipkati ročno. Za nekatere sisteme črtnih kod je način preverjanja standardiziran, lahko pa si ga zamislimo popolnoma po svoje.


9 0123456789

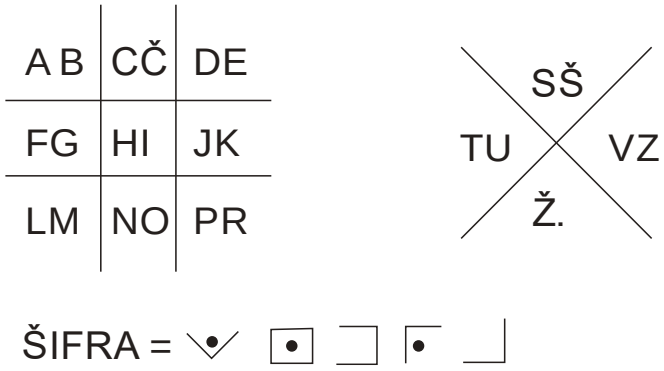
Šifriranje

Skrivna komunikacija je verjetno stara toliko kot človeštvo. Prve znane zapisane šifre so odkrili v Egiptu in so stare več kot štiri tisoč let. Skozi tisočletja so se razvili mnogi spodobni šifrirni sistemi. Njihova zapletenost je s časom naraščala, tako kot se je bogatilo znanje in sposobnost, da ljudje z najrazličnejšimi pripomočki šifre razbijemo. Danes se v elektronskih komunikacijah šifrirni sistemi intenzivno uporabljajo. Seveda so mnogo bolj kompleksni od teh, ki si jih bomo ogledali v nadaljevanju. Slonijo pa na enakih idejah.

Osnovno besedilo ali čistopis po nekem pravilu preoblikujemo v tajnopis. Poleg pravila postopek preoblikovanja določa tudi ključ. Pri šifriranju vedno predpostavimo, da je postopek šifriranja znan (prej ali slej se razve), razen pošiljatelja in prejemnika pa naj nihče ne bi poznal ključa.

Šifriranje z alternativno abecedo

Ena najbolj znanih alternativnih abeced je prostozidarska ali skavtska abeceda. Pravilo, po katerem se črke slovenske abecede preslikajo v črke prostozidarske abecede, je prikazano na spodnji sliki.



Črke razvrstimo v dve tabeli, kot kaže slika. Pri šifriranju posamezno črko nadomestimo s črtama, ki črko ločijo od ostalih črk, z izjemo sosede. Sosednji črki ločimo tako, da prvi narišemo samo črte, drugi pa dodamo še piko.

To šifro je zelo enostavno razbiti. Preštejemo pogostost posameznih znakov in jo primerjamo s pogostostjo znakov v običajnih besedilih. Najpogostejše znake bomo hitro izluščili, manj pogoste pa bomo lahko dobili iz besednih zvez. Daljše kot bo zašifrirano besedilo, bolj natančno bomo lahko rekonstruirali kodno tabelo.

Šifriranje z zamenjavami

Izmišljanje nove abecede je precej nerodno. Veliko raje pišemo znake, ki smo jih navajeni. Šifrirni sistem z zamenjavami so si zamislili že v starem Rimu v času Julija Cezarja. Po njem se tudi imenuje. Ideja je silno preprosta. Znake osnovne abecede zamenjamo z drugimi znaki

abecede, pri tem pa vrstnega reda znakov ne spremenimo. Rimljani so uporabljali šifrirno kolo

šifriramo pa lahko tudi s tabelo. V nadaljevanju bomo čistopis pisali z malimi črkami, tajnopis pa z velikimi. V spodnjem primeru je šifrirna abeceda zamaknjena za tri znake glede na osnovno abecedo. Ključ je torej 3 ali pa črka Č, s katero se začne šifrirna abeceda.

a	b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v	z	ž
Č	D	E	F	G	H	I	J	K	L	M	N	O	P	R	S	Š	T	U	V	Z	Ž	A	B	C

Šifriranje poteka enostavno. Znak iz čistopisa poiščemo v zgornji vrsti in namesto njega zapišemo znak tajnopisa iz spodnje vrste. Primer:

cezarjeva šifra → EHBČTMHAČ VLITČ

To šifro je zelo enostavno razbiti. V tajnopisu poiščemo najbolj pogosti znak. Najbolj pogosta črka v slovenščini je e, sledijo ji a, o in i. Zelo verjetno je da najbolj pogost znak v tajnopisu zamenjuje črko e. Kaj skriva tajnopis

ŽUŠHOS RČP MH ?

Boljšo šifro bi dobili, če bi črke poljubno premešali. V tem primeru si je ključ težje zapomniti. Pred slabimi 500 leti je Vigenere nadgradil Cezarjevo šifro tako, da je hkrati uporabil več Cezarjevih šifrirnih abeced. Ključ pri Vigenerejevi šifri je navadno kar beseda ali besedna zveza, recimo KRT. Če pri roki nimamo tablice z vsemi možnimi Cezarjevimi šifrirnimi abecedami, si pripravimo samo nujno potrebne – to so abecede, ki se začnejo s črkami K, R in T:

a	b	c	č	d	e	f	g	h	i	j	k	l	m	n	o	p	r	s	š	t	u	v	z	ž
K	L	M	N	O	P	R	S	Š	T	U	V	Z	Ž	A	B	C	Č	D	E	F	G	H	I	J
R	S	Š	T	U	V	Z	Ž	A	B	C	Č	D	E	F	G	H	I	J	K	L	M	N	O	P
T	U	V	Z	Ž	A	B	C	Č	D	E	F	G	H	I	J	K	L	M	N	O	P	R	S	Š

Z njo bomo zašifrirali zgoraj nerazkriti čistopis. Pod čistopis podpišemo ključ. Če je ključ prekratek, ga ponovimo. Črka ključa nam določa, katero Cezarjevo šifrirno abecedo uporabimo za šifriranje znaka čistopisa:

u	s	p	e	l	o		n	a	m		j	e
K	R	T	K	R	T		K	R	T		K	R
G	J	K	P	D	J		A	R	H		U	V

Zdaj so stvari bolj zapletene. Vigenerejev sistem je porušil strukturo besed. Hitro opazimo, da se ista črka čistopisa preslika v različne črke v tajnopisu. Obratno, ista črka tajnopisa lahko predstavlja različne črke v čistopisu. Šifro spet napademo s frekvenčno analizo, ki jo moramo delati za vsako črko ključa (Cezarjeve šifrirne abecede) posebej. Pri tem seveda ne vemo, kako dolg je ključ.

Šifriranje s premeščanjem

Gre za eno bolj enostavnih šifer. Po nekem pravilu znake premešamo med seboj. Dokler se je šifriralo in dešifriralo na roke, je bila šifra precej v uporabi. Zašifrirajmo sporočilo

antična šifra .

Temu sporočilu pravimo tudi čistopis. Pri premeščanju se čistopis največkrat napiše v matriko po vrsticah:

a n t i
č n a š
i f r a

preberemo pa ga po stolpcih:

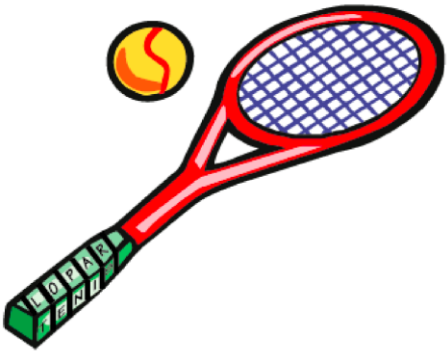
AČINNFTARIŠA .

Šifro lahko dodatno zapletemo, če stolpce premešamo med seboj, preberemo po diagonalah, v obliki spirale in podobno. To šifro je precej težko razbiti na roke. Ne moremo se zanašati na pogostost posameznih znakov, ampak moramo analizirati pogostost parov znakov. Za določitev števila znakov v vrstici pa nam prav pa pride tudi opazovanje razmerja med samoglasniki in soglasniki.

Šifriranje s premeščanjem so poznali že Špartanci. Čistopis so napisali na trak, trak ovili okrog palice in nato vsebino prebrali vzdolž palice.

Tudi Bobri se spopadajo s podobnimi problemi. V nalogah Teniški lopar in Trikotna šifra.

Teniški lopar



Igralci tenisa pogosto ovijejo ročaj svojega loparja, da ga bolje primejo. Včasih na oviti trak tudi kaj napišejo.

Eden od igralcev, ki je izgubil tekmo na Odprtem prvenstvu Bobrovije, je jezno odvil trak z ročaja svojega loparja in ga zalučal v travo. Sodnik je prebral napis na traku, **L T B P O E O E P N B T A I E E R S R R**, in brez težav ugotovil, kdo je bil ta igralec. Kdo?

Trikotne šifre

Bobrovka Beti si je izmislila sistem za pisanje skritih sporočil: črke sporočila napiše v trikotnik po vrsticah in jih prebere po stolpcih. Iz sporočila SKRITOSPOROČILO tako dobi SORIOKSOLRPČIOT.

S	K	R	I	T	O	S	P	O	R	O	Č	I	L	O

S	K	R	I	T
O	S	P	O	
R	O	Č		
I	L			
O				

S	O	R	I	O	K	S	O	L	R	P	Č	I	O	T

Svojima prijateljicama Ani in Cilki je poslala sporočilo PTTT?RAOOINRDAE. Kaj ji bosta odgovorili?

A. Obakrat.
B. Dvanajst.
C. V šolo.
D. Seveda.

Šifriranje z javnim in zasebnim ključem

Teorija pravi, da z daljšanjem ključa dobimo bolj varne tajnopise. Če je ključ dolg toliko kot čistopis, je šifrirni sistem popolnoma varen. Tudi Vigenerejev. To pa nas pripelje do še enega problema. Ključ morata poznati pošiljatelj in sprejemnik. Daljši kot je, težje si ga izmenjata in bolj nevarno je, da ga kdo prestreže.

Za izmenjevanje ključev se danes uporabljajo šifrirni sistemi z javnim in zasebnim ključem. Ti temeljijo na kompleksnosti postopkov - dešifriranje je mnogo bolj zapleteno od samega šifriranja. Danes se največ uporabljajo postopki, ki temeljijo na praštevilih. Iz dveh gromozanskih praštevil, ki jih računalniki dokaj hitro izračunajo, dobimo še večji produkt. Ta produkt je zelo težko razcepiti na obe praštevili in s tem rabiti šifro.

Danes se pri varni elektronski komunikaciji omenjajo certifikati. Gre za sistem javnih in zasebnih ključev, za katerimi se navadno skriva šifrirni sistem RSA. Ta je zasnovan ravno okrog ideje z velikimi praštevili. Javni ključ vsebuje samo produkt praštevil, ki ga je tako rekoč nemogoče razbiti na prafaktorja. Zasebni del ključa pa vsebuje prafaktorja. Šifriranje je zasnovano tako, da z javnim ključem lahko kdorkoli šifrira sporočila, odšifrira pa jih lahko samo lastnik, ki pozna prafaktorja. Na žalost pa je postopek šifriranja in dešifriranja (ob znanih prafaktorjih) preveč počasen. Zato se na ta način vedno prenesejo šifrirni ključi, nadaljevanje šifrirane komunikacije pa poteka po hitrih postopkih z enim samim ključem.

Bobri se v nalogah Cocsozšla tigsa in Golobi z dolgim nosom spopadajo s šifrirnimi sistemi z enim samim ključem. Znamo ugotoviti za katere šifrirne sisteme gre?

Cocsozšla tigsa

Bobrček si dopisuje s prijateljico vidro. Da njunih sporočil ne bi kdo prestregel in prebral, sta si domislila svojo šifro. Vsak soglasnik zamenjata z naslednjim soglasnikom po abecedi (B s C, C s Č in tako naprej, Ž pa zamenjata z B). Samoglasnike pustita pri miru.

Kako bi zapisala OB POL OSMIH OB POTOKU?

- × UB PUL USMOH UB PUTUA
- × UC RUM UŠNOJ UC RUVULA
- × OC ROM OŠNIJ OC ROVOLU
- × OŽ NOK ONLIG OŽ NOŠOJU

Golobi z dolgim nosom

Bober Matjaž pošilja prijateljici, bobrovki Alenki, razglednico s počitnic. Ker poštni golobi radi povsod vtikajo nosove, bo sporočilo skril. Napisal ji je

ŽAJTAMJOVTAJLIŠOPITAJROMZEVARDZOPEPEL

kar pomeni: Lepe pozdrave z morja ti pošilja tvoj Matjaž

Alenka mu je odgovorila

AKNELAOCITRAKAZALAVH

Kaj to pomeni?