Datu saspiešana I

Datu saspiešana ir datu pārveidošana kompaktā formā.

Datu saspiešanas pielietojumi:

- Arhivatori
- Attēlu, skaņas, video glabāšana un pārraide

Saspiešanas veidi:

- Bez zaudējumiem
  - Ziņojums = Ziņojums'
  - Teksts, datorprogrammas
- Ar zaudējumiem
  - Ziņojums ≈ Ziņojums'
  - Attēli, skaņa, video

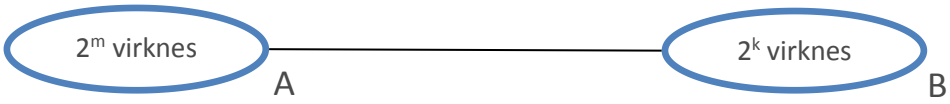
Saspiešana bez zaudējumiem

Universālās saspiešanas neiespējamība

Nav iespējama „universāla saspiešana", t.i. nav iespējams tāds algoritms, kas n bitu virkni saspiestu par k bitu virkni, kur k < n.

Kāpēc tā?

1 bits → 2 vērtības (0 vai 1)
m biti → 2^m vērtības
k biti → 2^k vērtības
k < m → $2^k < 2^m$



Kopā var A var atrast tādas virknes, kas saspiežas par vienu virkni kopā B. Atspiežot šīs virknes, nevar viennozīmīgi noteikt kāda ir bijusi sākotnējā virkne. Formālā pierakstā:

$\neg \exists x,y(x \neq y \ \& \ x \rightarrow z \ \& \ y \rightarrow z)$

Hofmana kodēšana

Pamatlietojums ir teksta un līdzīgas informācijas saspiešana. Piemēram, saspiešanas objekts varētu būt teksts latviešu vai angļu valodā.

Angļu valodā ir 96 simboli (ieskaitot dažādus speciālos simbolus, piemēram, cipari, punktuācijas simboli). Katru simbolu var kodēt ar 7 bitu virkni. Virkņu skaits – $2^7 = 128$. 128 > 96, līdz ar to katram simbolam var atrast 7 bitu virkni.

Izmantojot labāko saspiešanas algoritmu, apjomu var samazināt līdz 1,9 bitiem / simbols.

Teorētisks apakšējais novērtējums teksta saspiešanai: 1,3 biti / simbols.

Hofmana kodēšana sasniedz attiecību 4,7 biti / simbols.

Prefiksu kodēšana

Hofmana kodēšana ir speciālgadījums prefiksu kodēšanai.

Jebkura saspiešanas metode balstās uz likumsakarībām datos.

Prefiksu kodēšanā izmantotā likumsakarība: dažādi burti sastopami ar dažādu biežumu.

Burts	Biežums
E	11,2%
A	8,5%
R	7,6%
I	7,5%
Q	0,19%
J	0.19%
Z	0.27%
X	0,24%

Tabula 1: 4 visbiežāk un 4 visretāk sastopamie simboli angļu valodā.

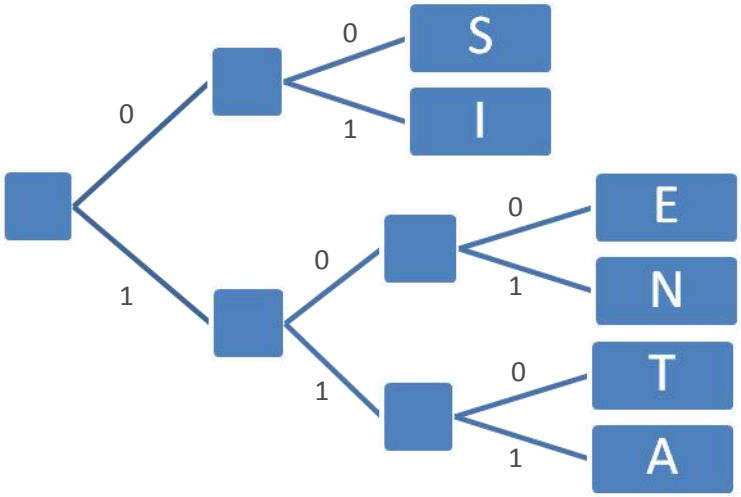
Ja katru burtu kodē ar 7 bitiem, tad arī reti sastopamie burti tiek kodēti ar 7 bitiem. Piemēram,

E → 0110110
Q → 1011001

Prefiksu kodēšana atsakās no šī principa.

- Bieži sastopamus burtus kodējam ar mazāku bitu skaitu. Piemēram,
  - E → 5biti
- Reti sastopamus – ar lielāku bitu skaitu. Piemēram,
  - Q → 12biti

Piemēram zemāk dots prefiksu koks.



Ilustrācija 1: Prefiksu koks

Izmantojot doto prefiksu koku, iespējams kodēt simbolu virknes, kas sastāv no simboliem ‘S’, ‘I’, ‘E’, ‘N’, ‘T’ un ‘A’.

Piemēram, virkne ‘TESTS’ tiek kodēta sekojoši:
T → 110
E → 100
S → 00
T → 110
S → 00
TESTS → 11010000110

Atkodēšana

Lai atkodētu virkni, kas kodēta izmantojot prefiksu koku, sāk koka saknē un virzās lejā pa koka šķautnēm, katrā mezglā izvēloties virzienu, ko norāda kārtējais bits virknē, līdz sasniegta koka lapa. Iegūts viens virknes simbols. Atkārtojot procedūru, atkal sākot saknē kamēr virknē ir biti, var iegūt sākotnējo virkni.

1. piemērs. Atkodēt virkni ‘11100110100’, izmantojot augstāk doto prefiksu koku.

    111 → A
    00 → S
    110 → T
    100 → E

    ASTE

2. piemērs. Atkodēt virkni ‘0001100101111’, izmantojot augstāk doto prefiksu koku.

    00 → S
    01 → I
    100 → E
    101 → N
    111 → A

    SIENA

Daži jautājumi par prefiksu kodēšanu.

1. Kāpēc termins „prefiksu kodēšana”?

Korektāk būtu „bezprefiksu kodēšana”.

Virkne a ir b prefikss, ja b = ac, c – kaut kāda virkne.

Lietojot prefiksu koku, neviena no bitu virknēm, ko izmanto kodēšanā, nav citas virknes prefikss. Šī īpašība piemīt jebkuram prefiksu kokam.

2. Kāpēc tas ir svarīgi?

Pieņemsim, ka A, B tiek kodēti ar virknēm, kas ir viena otras prefikss.

A→ 00
B → 000
Ko nozīmē virkne „000000”? Iespējami varianti BB un AAA.

Ja viena virkne ir citas virknes prefikss, viennozīmīga datu atspiešana nav iespējama. Viennozīmīgai atspiešanai nepieciešams, lai neviena virkne nav citas virknes prefikss.

Hofmana kodēšana

- **Speciālgadījums prefiksu kodēšanai**.
- **Algoritms prefiksu koka ģenerēšanai** - Hofmana kodēšana ir metode kā uzģenerēt prefiksu koku, ja zināma katra simbola parādīšanās biežums.
- **Metode dod optimālu prefiksu koku** - dodot algoritmam pareizos burtu biežumus, metode dod tādu prefiksu koku, kuru izmantojot virknes kodēšanā, iegūst īsāko iespējamo virkni.

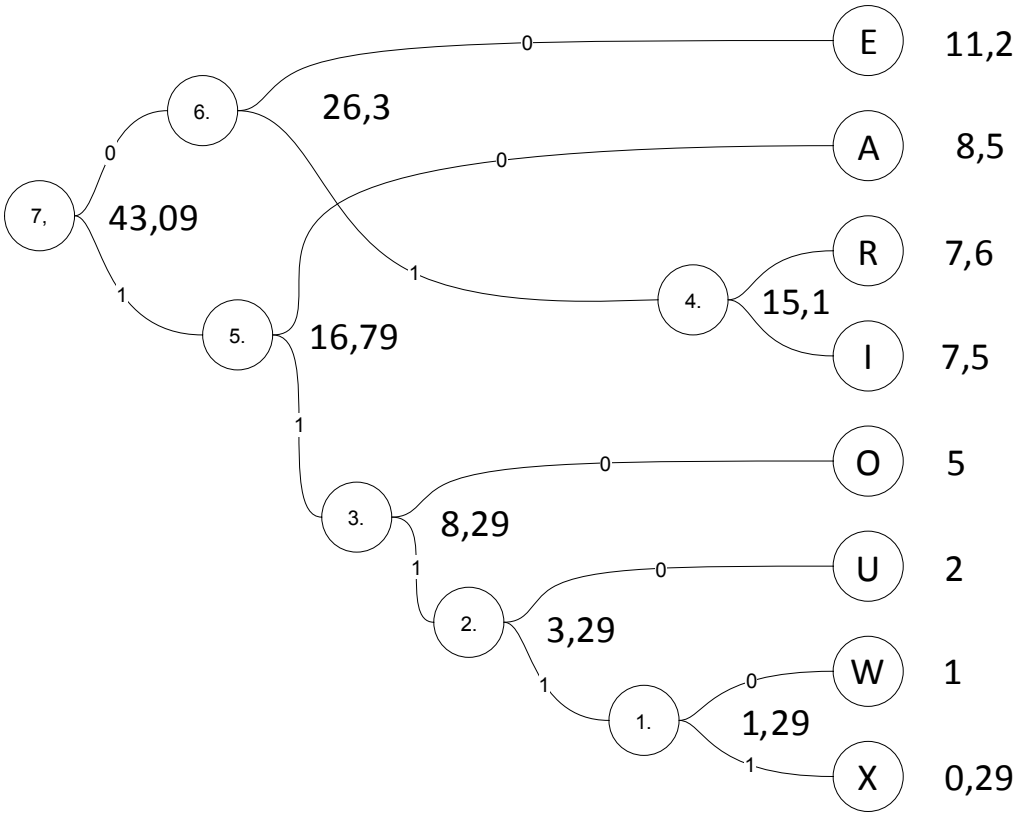
Burts	Biežums
E	11,2%
A	8,5%
R	7,6%
I	7,5%
U	5%
O	2%
W	1%
X	0,29%

Tabula 2: Daži burtu biežumi

Hofmana algoritms

- Sāk ar n kokiem, katram no kuriem ir 1 lapa, kas atbilst 1 burtam.
- Atkārto, kamēr ir nesavienoti koki
    1. Atrod 2 kokus ar mazākajiem biežumiem (X, Y, biežumi P_x, P_y)
    2. Izveido koku: sakne Z (jauna virsotne). Šķautnes ZX, ZY. Biežums $P_z = P_x + P_y$
- Katrām divām šķautnēm, kas iziet no virsotnes, patvaļīgi piešķirt apzīmējumus 0, 1.

Piemēram, doti 8 koki, katrs sastāv no vienas lapas, kas atbilst vienam burtam no iepriekš dotās tabulas. Katram kokam ir piekārtots atbilstošā burta parādīšanās biežums.



Ilustrācija 2: Hofmana koks dotajiem burtu biežumiem

Izpildot aprakstīto algoritmu iegūst Hofmana koku (skat. Ilustrācija 2: Hofmana koks dotajiem burtu biežumiem).

Atliek katrām divām šķautnēm, kas iziet no virsotnes, patvaļīgi piešķirt apzīmējumus 0, 1.

Koka iegūšana pa soļiem:

1. Mazākie biežumi ir kokiem W un X. Izveido jaunu koku 1., kuram P= $P_W + P_X = 1,29$.
2. Mazākie biežumi ir kokiem 1. un U. Izveido jaunu koku 2., kuram P = $P_1 + P_U = 3,29$.
3. Mazākie biežumi ir kokiem 2. un O. Izveido jaunu koku 3., kuram P = $P_2 + P_O = 8,29$.
4. Mazākie biežumi ir kokiem R un I. Izveido jaunu koku 4., kuram P = $P_R + P_I = 15,1$.
5. Mazākie biežumi ir kokiem 3. un A. Izveido jaunu koku 5., kuram P = $P_3 + P_A = 16,79$.
6. Mazākie biežumi ir kokiem 4. un E. Izveido jaunu koku 6., kuram P = $P_4 + P_E = 26,3$.
7. Pēdējie nesavienoti palikuši koki 5. un 6. Izveido jaunu koku 7., kuram P = $P_5 + P_6 = 43,09$.