TÖL403G GREINING REIKNIRITA

11. Gráðug reiknirit 2

Hjálmtýr Hafsteinsson
Vor 2022

Í þessum fyrirlestri

- Huffman kóðar (*Huffman codes*)
  - Gagnaþjöppun með kóðum
  - Forstrengsfríir kóðar (*prefix-free codes*)

- Almennt um gagnaþjöppun
  - Notkun á Huffman kóðun
  - Aðrar þjöppunaraðferðir

4.4

Einfalt dæmi um þjöppun

- Skrá hefur 100.000 bókstafi af aðeins 6 mismunandi gerðum

Stafur	Tíðni
a	45%
b	13%
c	12%
d	16%
e	9%
f	5%

Ef stafirnir eru geymdir á 8-bita ASCII formi þá þarf 8*100.000 = 800.000 bita fyrir skránna

Við gætum reyndar komist af með 3 bita per staf, þar sem það eru aðeins 6 ólíkir stafir

Til dæmis: *a*: 000, *b*: 001, o.s.frv.

Þá þarf aðeins 300.000 bita fyrir skránna

Þetta er það besta sem við getum gert með föstum fjölda bita per staf

En hvað með að nota mismunandi marga bita eftir því hversu algengir stafirnir eru?

Betri kóðun, dæmi

Notum færri bita fyrir algenga stafi og fleiri bita fyrir sjaldgæfa stafi

Stafur	Kóði	Tíðni*Lengd	Meðallengd
a	0	0.45*1	0.45
b	101	0.13*3	0.39
c	100	0.12*3	0.36
d	111	0.16*3	0.48
e	1101	0.09*4	0.36
f	1100	0.05*4	0.20

Samtals: **2.24** bitar að meðaltali

Hér þurfum við því aðeins 224.000 bita fyrir skránna

Breytileg kóðalengd krefst þess að kóðinn sé forstrengsfrír (*prefix-free*)

Kóði er forstrengsfrír ef enginn stafakóði er forstrengur fyrir annan stafakóða

Í þessu dæmi þá hefur *a* kóðann 0, það þýðir að enginn annar kóði getur byrjað á 0

Forstrengsfríir (*prefix-free*) kóðar

- Kóðar af fastri lengd eru alltaf forstrengsfríir
  - Það er engin leið fyrir einn stafakóða að vera forstrengur annars, nema þeir séu sami kóðinn!

- Kóðar af breytilegri lengd verða að vera forstrengfríir
  - Annars er ómögulegt að afkóða þá

Stafur	Kóði
a	0
b	101
c	100
d	111
e	1101
f	1100

Til dæmis: Ef stafurinn *b* hefði haft kóðann 01 í dæminu á undan

Fáum bitarununa 0100101

Er fyrsti bitinn (0) afkóðaður sem *a*, eða
eru fyrstu tveir bitarnir (01) afkóðaðir sem *b*?

Eins og kóðinn er í dæminu þá á að afkóða þetta sem *acb* (0 100 101)

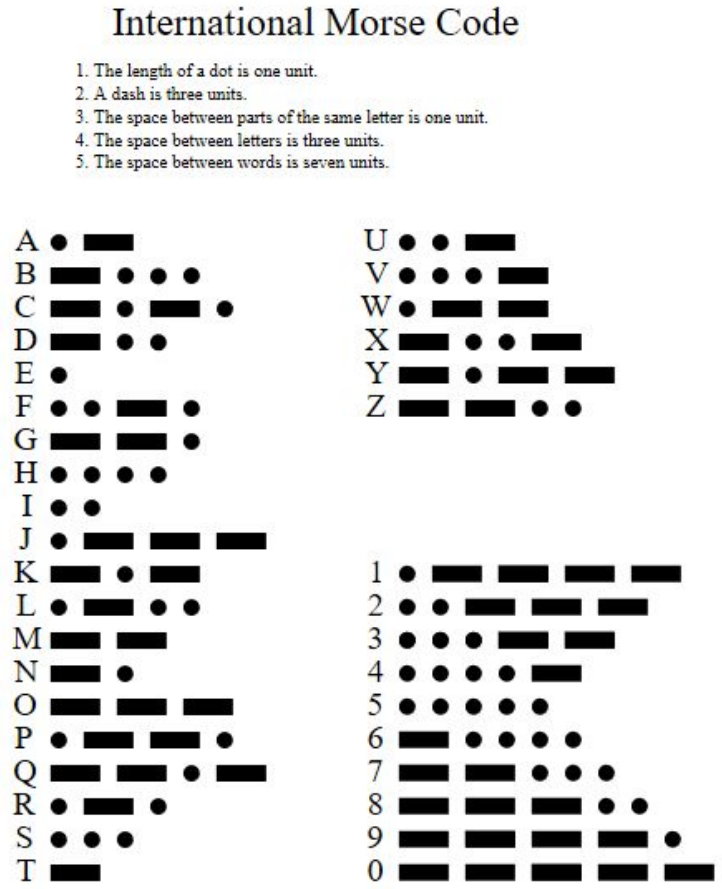
Morse kóði

Fyrst notaður árið 1844, staðlaður 1865

- Gamalt dæmi um kóða af breytilegri lengd
  - Algengir stafir (E, T, I, ...) hafa stuttan kóða
- Hann virðist ekki vera forstrengsfrír
  - Kóðinn fyrir E (●) er forstrengur fyrir marga stafi, A, H, I, ...
- En í raun er Morse kóði forstrengsfrír þríundarkóði
  - Það verður alltaf að vera eyða á milli stafanna
  - Eyðan (□) er því þriðja táknið

Kóðinn fyrir E er því ●□ og er ekki forstrengur fyrir A: ●▬□

Til að auðvelda afkóðun eru settar tvær eyður (□□) á milli orða



"Greining reiknirita" í Morse kóða:

UTF-8 (*Unicode Transformation Format*)

- Hannaður af Ken Thompson og Rob Pike
  - Tilgangur: Þurfa ekki að nota 2 eða 4 bæti fyrir algeng ensk tákn
  - Getur táknað öll (~1.1 milljón) Unicode táknin með 1 til 4 bætum

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

Ef kóðinn er aðeins eitt bæti þá er fyrsti bitinn **0**-biti

Annars byrjar fyrsta bætið í kóðanum alltaf á **11**

Næstu bitar á eftir þeim segja til um hversu mörg bætin eru:

0: tvö bæti

10: þrjú bæti

110: fjögur bæti

Framhaldsbætin byrja alltaf á **10**

Þau eru einu bætin sem byrja á **10**

Nokkuð stór hluti bitanna er notaður til að tryggja kóðinn sé forstrengsfrír

Ef 4 bæti: 21 gagnabiti og 11 kóðabitar
þ.e. aðeins ~ 66% bitanna eru gagnabitar

Besti kóðinn

- Hvernig búum við til besta kóðann miðað við gefna stafatíðni?

Með gráðugu reikniriti!

David Huffman fann þetta reiknirit árið 1951

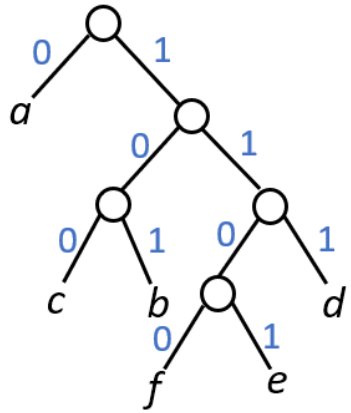
Hann var nemendi í námskeiði í MIT og þetta var verkefni í námskeiðinu

Gróf lýsing reikniritsins: Sameina tvö sjaldgæfustu táknin í eitt tákn og halda áfram endurkvæmt þar til aðeins eitt tákn eftir

Getum táknað forstrengsfría kóða sem tvíundartré:

- Stafirnir eru í laufunum
- Hver leggur er merktur með 0 eða 1
- Vegurinn frá rót í lauf er þá kóði þess stafs

6-stafa kóðinn frá fyrri glæru:

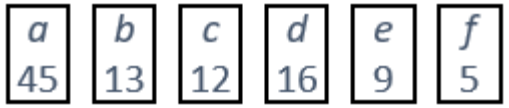


Engir stafir í innri hnútum, annars væri kóðinn ekki forstrengsfrír

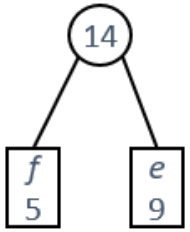
Einfalt sýnidæmi

Í upphafi er hver stafur sjálfstætt tré, sameinum svo eitt og eitt tré í einu

6 tré

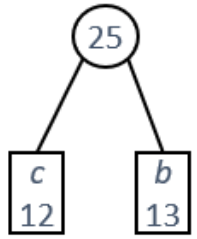


Sameinum tré með lægstu tíðnir, *f* og *e*:

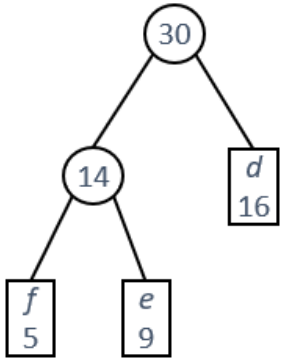


Nú eru 5 tré, með tíðnir 45, 13, 12, 16 og 14

Sameinum tré með lægstu tíðnir, *c* og *b*:

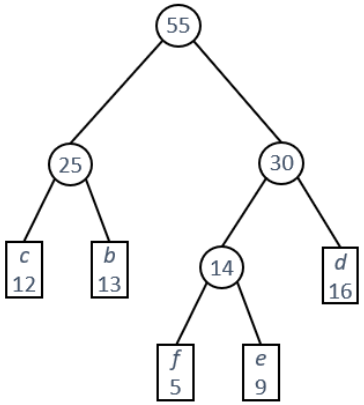


Sameinum næst *d* og tréð með tíðnina 14:

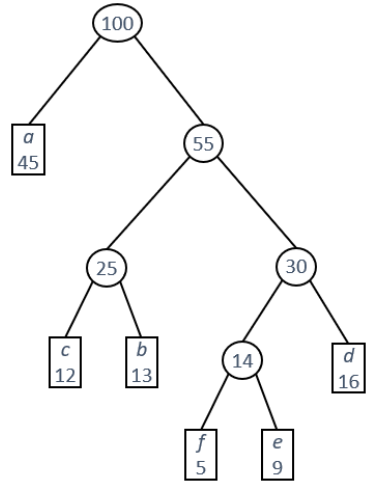


Nú eru 3 tré eftir, með tíðnir 45, 30 og 25

Sameinum tré með tíðnir 25 og 30:



Endum svo með:



Huffman æfing

- Höfum 3 tákn: *a*, *b*, *c* og þrjár ólíkar kóðanir fyrir þau. Eru þær Huffman kóðanir? Ef svo, teiknið kóðunartréð. Annars, af hverju ekki?

- *a*: 0, *b*: 10, *c*: 11

- *a*: 0, *b*: 1, *c*: 00

- *a*: 10, *b*: 01, *c*: 00

Stærra sýnidæmi

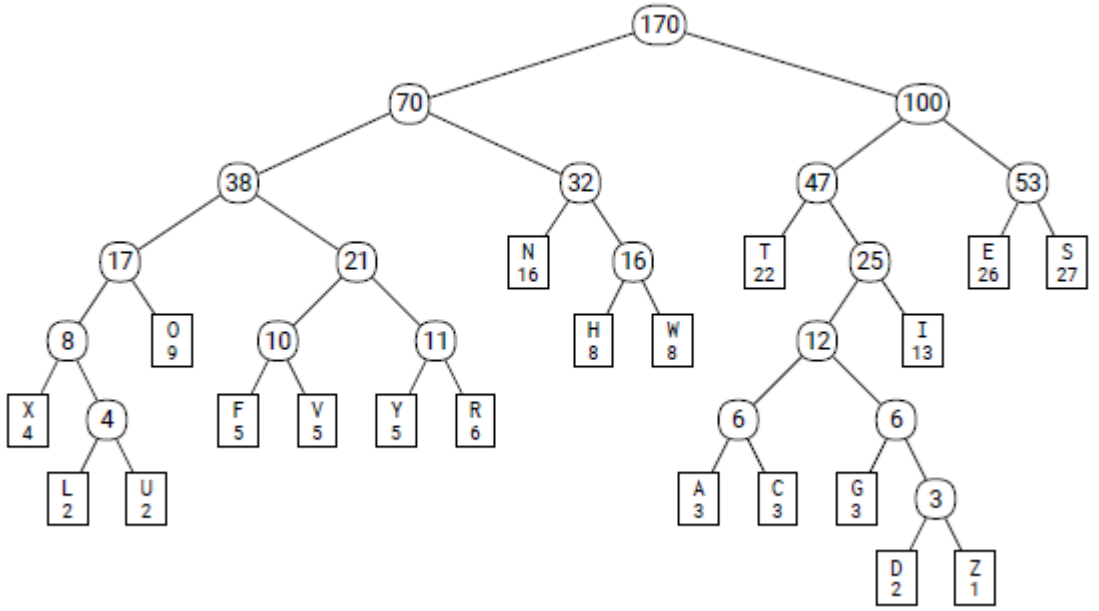
Byrjum með textann:

THISSENTENCECONTAINSTHREEASTHREECSTWODSTWENTYSIXESFIVEFST
HREEGSEIGHTHSTHIRTEENISTWOLSSIXTEENNSNINEOSSIXRSTWENTYSEV
ENSSTWENTYTWOTSTWOUSFIVEVSEIGHTWSFOURXSFIVEYSANDONLYONEZ^{6}

Tíðni bókstafa í honum:

A	C	D	E	F	G	H	I	L	N	O	R	S	T	U	V	W	X	Y	Z
3	3	2	26	5	3	8	13	2	16	9	6	27	22	2	5	8	4	5	1

Keyrum reiknirit Huffman og fáum kóðunartréð:



Hér er stysti kóðinn 3 bitar (N, T, E, S)

Lengsti kóðinn er 7 bitar (D, Z)

Textinn hefur 170 bókstafi
Kóðinn er 649 bitar (~3.8 bitar per staf)

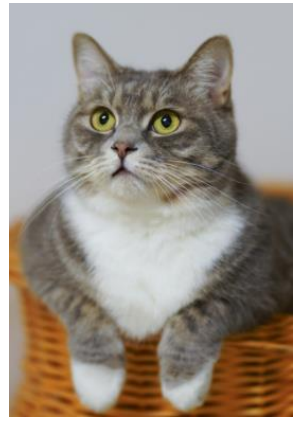
Tímaflækja

- Geymum táknin (trén) í forgangsbiðröð
  - Finna tvær lægstu tíðnirnar kostar þá $O(\log(n))$
- Ef við byrjum með tóma forgangsbiðröð, þá
  - þurfum við $2n-1$ Innsetningar (*Insert*)
  - og $2n-2$ EyðaMinnsta (*ExtractMin*)
- Heildartíminn er þá $O(n \log(n))$

```
BuildHuffman(f[1..n]):
  for i ← 1 to n
      L[i] ← 0;  R[i] ← 0
      Insert(i, f[i])
  for i ← n to 2n − 1
      x ← ExtractMin( )      ⟨⟨find two rarest symbols⟩⟩
      y ← ExtractMin( )
      f[i] ← f[x] + f[y]     ⟨⟨merge into a new symbol⟩⟩
      Insert(i, f[i])
      L[i] ← x;  P[x] ← i    ⟨⟨update tree pointers⟩⟩
      R[i] ← y;  P[y] ← i
  P[2n − 1] ← 0
```

- Getum gert kóðun/afkóðun með trénu á $O(m)$ tíma, þar sem m er lengd skilaboðanna

Gagnaþjöppun



65.784 bæti

1.998 bæti

~33 föld þjöppun

- Tapandi (*lossy*) þjöppun
  - Þjappar gögnum með því að einfalda þau og geyma nálgun á þau (*approximation*)
  - Getur náð mjög mikilli þjöppun (10-100 föld) án mikillar sjáanlegrar bjögunar
  - Ekki hægt að ná aftur upphaflegu gögnunum (*irreversable*)
  - Gerð þjöppunar fer eftir tegund gagna: myndir, tónlist, myndskeið, ...

Dæmi: JPEG, MP3, MPEG, ...

- Taplaus (*lossless*) þjöppun
  - Nýtir sér tölfræðilega umfremd (*redundancy*) til að tákna gögn á styttri máta
  - Nær yfirleitt ekki mjög mikilli þjöppun (dæmigert 2-3 föld)
  - Þjöppunin er viðsnúanleg, við fáum aftur nákvæmlega sömu gögnin
  - Oftast frekar almennar aðferðir sem virka á allar gerðir gagna

Dæmi: Huffman, LZ, TIFF, PNG, ...

Áhugaverð þjöppunarreiknirit

Bæði "orðasafns"-reiknirit (*dictionary coders*)

- Tvö þekktustu taplausu reikniritin eru LZ77 og LZ78
  - Sett fram af Abraham Lempel og Jacob Ziv árin 1977 og 1978

LZ77 notar hliðrandi glugga, finnur endurteknar gagnablokkir og setur í staðinn tilvísun á fyrra tilvik

LZ78 býr jafnóðum til orðasafn og setur tilvísun í hana í stað endurtekinna gagnablokka

- Notuð mjög víða í þjöppunarforritum
  - GIF-skrár, ZIP-skrár, ýmis netbúnaður, ...

- LZW þjöppunin
  - Terry Welch bjó til sniðuga útfærslu á LZ78
  - Fyrirtækið hans, Sperry Corp. sótti um einkaleyfi á LZW árið 1983
  - LZW var notað í GIF myndaforminu og Unisys (sem keypti Sperry) fór að rukka fyrir það!
  - Varð til þess að PNG myndaformið var skilgreint
  - Einkaleyfið féll úr gildi árið 2002 (20 ára gildistími)

Notkun á Huffman kóða

- Galli við Huffman kóðun:

  Þurfum að vita tíðnina á öllum stöfunum til að búa til kóðann

  - Hentar þá vel á eftir annari þjöppun (teljum fjölda tákna um leið!)

Í **`gzip`** er Huffman kóðun beitt á eftir LZ77 aðferðinni

Í **`bzip2`** er Huffman kóðun beitt á eftir Burrows-Wheeler aðferðinni

Huffman kóðun er lokaþrep í JPEG, MP3 og PNG

Nær 20 − 40% aukaþjöppun

Oftast um 20% aukaþjöppun

Huffman kóðun notuð til að "kreista út" alla mögulega umfremd (*redundancy*)

Kvik (*adaptive*) Huffman kóðun

- Tveir gallar á venjulegri Huffman kóðun:
  - Þurfum tvær umferðir (til að safna tíðnigildum)
  - Þurfum að senda kóðunartréð með þjöppuðu gögnunum til að hægt sé að afkóða þau

Minnkar þjöppunina, sérstaklega ef gagnamagnið er lítið

- Lausn: Kvik Huffman kóðun

  Búum til kóðunartréð jafnóðum

  Á hverjum tíma er tréð Huffman tré fyrir þá stafi sem hafa sést hingað til

  Þegar við fáum nýjan staf þá notum við tréð til að kóða stafinn og uppfærum svo tréð út frá nýrri tíðni hans

Oftast ekki eins góð þjöppun og venjuleg Huffman

En ef tíðni stafa tákna er breytileg í stórri skrá þá getur Kvik Huffman kóðun verið betri

Fyrirlestraæfingar

1. Ef við höfum 16 mismunandi tákn, sem öll hafa nákvæmlega sömu tíðni, hvaða kóðalengd gefur þá Huffman kóðun fyrir þau?

2. Búið til Huffman kóða fyrir táknin *a*:20, *b*:10, *c*:5, *r*:8 og sýnið kóðann fyrir strenginn "abraca"

3. Við höfum 1000 stafa streng. Hversu mikið er hægt að þjappa honum?