Algoritmusok és adatszerkezetek II
Tömörítés témakör jegyzete

Készült Ásványi Tibor előadásai és gyakorlatai alapján

Sárközi Gergő, 2021-22-1. félév
Nincsen lektorálva!

Tartalomjegyzék

1. Tömörítés

- csak veszteségmentessel foglalkozunk
- kód = kódszavak nemüres halmaza (kódfával ábrázolható)
- betűnkénti kódolás: adatot betűnként $\sum \to \mathbb{C} \subset \mathbb{T}^*$ bijektív (kölcsönösen egyértelmű) leképezéssel készítjük
  - $\sum$ = ábécé, karakterek halmaza
  - $\mathbb{C}$ = kód
  - $T = \{0, 1\}$ általában
  - továbbiakban legyen $r = |\mathbb{T}|$ és $d = |\sum|$

2. Naiv módszer

- egyenletes kód: kód szavainak hossza egyenlő
- egy karakter (min) $\lceil log_r d \rceil$ hosszan kódolható
- kódtáblázat: karakter-kód párok
- kódtáblázatot is csatolni kell az adattal (és így mérni a tömörítés effektivitását)

2.1. Példa

Bemenet: ERRE_ARRA_MERRE_ARRA
$\sum = \{E, R, \_, A, M\} \implies d = 5$
$l = \lceil log_2 5 \rceil = 3$
Kimenet hossza: 20x3=60 + kódtáblázat

Betű	**Kód**
E	000
R	001
_	010
A	011
M	100

3. Huffmann-kód

- betűnkénti optimális kódolás
- 2 hiba: 2x kell beolvasni és betűnkénti kódolás (ami pl. képeknél nem nagyon tud tömöríteni)
- nem egyértelmű: azonos gyakoriságú betűk (és 0-1) felcserélhetők
- prefix-kód: kódszavak halmaza prefix mentes
- kódtáblázatot is csatolni kell az adattal
- kódfa: szigorúan bináris fa: 2d-1 csúcs

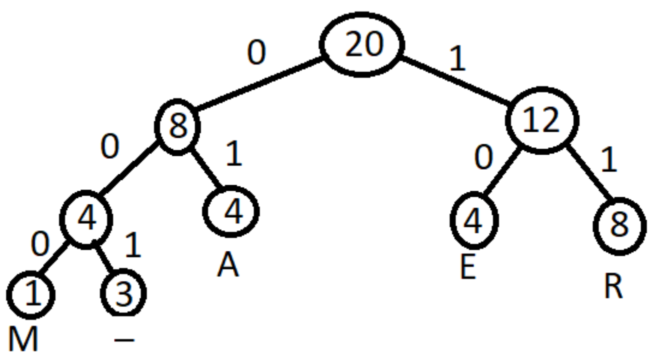
3.1. Példa

Bemenet: ERRE_ARRA_MERRE_ARRA

Betű	**Gyakoriság**
E	4
R	8
_	3
A	4
M	1

minPrQueue
<(1,M),(3,_),(4,A),(4,E),(8,R)>
<(4,M_),(4,A),(4,E),(8,R)>
<(4,E),(8,R),(8,M_A)>
<(8,M_A),(12,ER)>
<(20,M_AER)>

- amíg a sorban van ≥ 2 elem
  - szedjük ki őket, készítsünk egy szülő node-ot nekik (bal/jobb oldalt kapja egy gyerek a gyakorisága alapján)
  - rakjuk be a szülő node-ot a gyerekek értékének összértékével
- kész a fa



TODO stuki (kóddal együtt - kóddal tesztelve)

4. Lempel-Ziv-Welch (LZW) módszer

- nem betűnkénti kódolás: szótárkód
  - nincs meg az a hatékonysági határ, mint Huffmann kódnál
- lépésről lépésre egy S kódtáblát (szótárat) bővít
- szótár tulajdonságai:
  - egybetűs szavak alapból szerepelnek benne
  - ha egy szó benne van a szótárban, akkor minden kezdődarabja is benne van
  - minden tárolt szónak ($x \in S$) fix hosszúságú kódja van ($c(x)$)
- csak egyszer olvassuk be az adatot: egy időben kódolunk és építünk szótárat
- kódtáblát nem kell csatolni a kódolt adathoz
- kódolás: ha $x \in S$ szót találunk és a következő betű, Y, már $\notin S$, akkor $c(x)$-et kiírjuk, xY szót felvesszük a S szótárba. $c(xY)$ a legkisebb szabad kód. A beolvasást az Y betűvel folytatjuk.
- $c(x)$ fix hosszú, méghozzá ez a hossz egy előre ismert konstans (általában 12 bit)
- a kódolt adat mellé nem szükséges a szótár csatolása
- hibák:
  - hosszú szöveg esetén túl sok új kód van bevezetve, ami lassít
  - ezért korlátozhatjuk a kódszavak számát, hosszát
  - vagy csak a bemenet egy kezdőszeletén építjük a szótárat, utána már csak használjuk
- Kódot egyből felhasználjuk miután berakjuk szótárba: nem probléma, így is ki tudjuk találni a kód utolsó karakterét: azonos az elsővel

4.1. Példa

Bemenet: BBCABCABABAAC			
Ki kód	Akt. szó	Köv. szó	Új kód
2	B	B	4
2	B	C	5
3	C	A	6
1	A	B	7
5	BC	A	8
7	AB	A	9
9	ABA	A	10
1	A	C	11
3	C	-	-

1. ábra. Kódolás példa

Bemenet: 2 2 3 1 5 7 9 1 3			
Be kód	Akt. szó	Köv. szó	Új kód
2	B	B	4
2	B	C	5
3	C	A	6
1	A	B	7
5	BC	A	8
7	AB	A	9
9	ABA	...	10
...	...	...	...
...	...	...	...

2. ábra. Dekódolás példa

4.2. Stuki

- Ha a kódok b bitesek, akkor $MAXCODE = 2^b - 1$
- Item: szó-kód pár ($string : \sum <>$ és $code : \mathbb{N}$ pár)