Poročilo tretje laboratorijske vaje pri predmetu Informacija in kodi

Rok Prezelj

Univerza v Ljubljani, Fakulteta za elektrotehniko
E-pošta: rp0067@student.uni-lj.si

Povzetek. *V laboratorijski vaji smo implementirali LZW dekodirnik, preverili pravilnost delovanja z MD5 izvlečki ter ocenili učinkovitost kompresije na različnih vrstah datotek. Analizirali smo tudi kompresijo pri delnem kodiranju besedila.*

1 Uvod

Algoritem LZW (Lempel–Ziv–Welch) je klasičen brezizgubni postopek kompresije, ki izkorišča ponavljajoče se vzorce v podatkovnih tokovih. Osnovna ideja je izgradnja slovarja podnizov, ki se med kompresijo dinamično razširja, med dekompresijo pa se reproducira iz kodnih mest [2].

Cilj vaje je implementirati dekodirnik LZW ter z njim dekompresirati datoteke, ki jih predloga kode najprej komprimira. S pomočjo MD5 izvlečkov preverimo pravilnost delovanja, nato pa izračunamo teoretično velikost kompresiranega sporočila in ovrednotimo učinkovitost kompresije.

2 Metodologija

2.1 Opis algoritma LZW

LZW deluje po naslednjih korakih:

1. Slovar inicializiramo z vsemi 256 možnimi bajti.
2. Med kompresijo bere vhodno besedilo in išče najdaljši podniz, ki že obstaja v slovarju.
3. Ko se najdaljši podniz zaključi, v izhod zapišemo njegovo kodno mesto, nato v slovar dodamo razširjen podniz.
4. Kodno zaporedje predstavlja kompresirano sporočilo.

Pri dekompresiji je potrebno slovar rekonstruirati identično kot pri kompresiji. Poseben primer nastopi, ko se pojavi kodno mesto, ki še ne obstaja v slovarju; v tem primeru se določi iz prejšnjega vnosa v skladu z LZW pravilom $w + w[0]$.

2.2 Implementacija dekodirnika

Implementiran dekodirnik inicializira slovar z enobajtnimi zapisi ter nato zaporedno obnavlja podatkovno vsebino:

- če kodno mesto obstaja v slovarju, ga preberemo,
- sicer uporabimo posebno pravilo za primere $KwKwK$,
- v slovar dodamo $w + c$, kjer je w prejšnje dekodirano geslo, c pa prvi bajt trenutno dekodiranega niza.
- celoten izhod združimo v binarno datoteko.

Pravilnost implementacije smo preverili z MD5 izvlečki.

2.3 Ocenjevanje gospodarnosti kompresije

V članku [5] avtorji ne podajo izrecne enačbe za kompresijsko razmerje algoritma LZW, vendar pa se sklicujejo na uveljavljeno teoretično oceno iz [5], ki določa pričakovano število LZW-fraz v zaporedju dolžine n. Po tej klasični analizi velja:

$$M_n \approx \frac{n}{H},$$

kjer je H entropija informacijskega vira (v bitih na simbol), M_n pa pričakovano število fraz, ki jih generira LZW. Avtorji to oceno uporabijo kot osnovo za primerjavo med standardnim LZW in njihovo nadgrajeno, na napake odporno različico.

Ker LZW vsako frazo zapiše s kodnim mestom širine b bitov, je teoretična velikost kompresiranega zaporedja približno:

$$\text{velikost}_{\text{LZW}} \approx M_n \cdot b \approx \frac{n}{H} \cdot b.$$

V kontekstu naše naloge privzamemo, skladno s specifikacijo GIF-formata in klasičnimi izvedbami LZW [5], da je širina kodnih mest stalna in enaka:

$$b = 12 \text{ bitov}.$$

Zato dobimo poenostavljeno oceno teoretične velikosti LZW zapisa:

$$\text{velikost}_{\text{LZW}} \approx \frac{12n}{H} \text{ bitov}, \qquad \text{oz.} \qquad \frac{12n}{8H} \text{ bajtov}.$$

Ta ocena predstavlja idealizirano kompresijsko razmerje LZW in ni neposredno primerljiva z dejansko velikostjo tekstovne datoteke, v kateri so kodna mesta shranjena kot desetiška ASCII števila.

3 Rezultati

3.1 Preverjanje pravilnosti dekompresije

Za vse štiri uporabljene datoteke se MD5 izvlečka popolnoma ujemata, kar potrjuje pravilnost implementacije dekodirnika:

- `besedilo.txt`: `727c3bd61031d70c31549a0ca27eb876`
- `posnetek.mp3`: `be9d13947f025222b3485040a69ddc86`
- `slika.bmp`: `d1a0bf3f1f31040736151e9fef04e8fa`
- `lab3.zip`: `b8d94e76fbc608cae9e3784163395402`

3.2 Preizkus kompresije na različnih vrstah datotek

Datoteka	Velikost	LZW	Razmerje
besedilo.txt	1 038 950	304 229	3.415
posnetek.mp3	5 255 489	3 761 891	1.397
slika.bmp	1 839 122	1 011 110	1.819
lab3.zip	1 936 792	1 474 977	1.313

Tabela 1: Rezultati kompresije LZW za štiri različne vrste datotek. Enote so bajti.

Interpretacija rezultatov:
Besedilna datoteka doseže najvišjo stopnjo kompresije, saj vsebuje veliko redundance in ponavljajočih se vzorcev. Surova BMP slika se prav tako dobro komprimira zaradi izrazite strukture in ponovitev v slikovnih podatkih. Nasprotno pa ZIP arhiv in MP3 datoteka, ki sta že predhodno kompresirana, dosežeta bistveno slabše rezultate, saj v njih ostane zelo malo dodatne redundance, ki bi jo lahko LZW izkoristil.

3.3 Delno kodiranje besedila

Za besedilo smo izračunali ocenjene velikosti LZW kompresije pri delih vhodne datoteke od 10% do 100%.

Delež	Original (B)	LZW (B)	Razmerje
10%	103 895	41 781	2.487
20%	207 790	75 939	2.736
30%	311 685	107 363	2.903
40%	415 580	137 886	3.014
50%	519 475	167 480	3.102
60%	623 370	195 735	3.185
70%	727 265	223 451	3.255
80%	831 160	251 108	3.310
90%	935 055	277 770	3.366
100%	1 038 950	304 229	3.415

Tabela 2: Učinkovitost LZW kompresije pri delnem kodiranju besedila.

Opazujemo:

- več kot beremo besedilo, več vzorcev se nauči LZW slovar,
- rast učinkovitosti je monotona,
- pri 40–60% se slovar že zelo dobro stabilizira,
- največje razmerje (3.415) dosežemo pri celotnem besedilu.

3.4 Zakaj je datoteka `compressed_*.txt` večja?

Čeprav je teoretična velikost LZW-kompresije razmeroma majhna (na primer približno 304 kB za 1 MB besedila), je dejanska datoteka `.txt`, ki vsebuje kodna mesta, bistveno večja in lahko v našem primeru doseže tudi 1.4 MB. Razlog za to je, da so kodna mesta shranjena v obliki ASCII števil, kot so na primer nizi `"1234,567,..."`, pri čemer vsak znak zasede en bajt. Poleg tega so v zapisu vključene vejice in morebitni presledki, ki dodatno povečajo velikost datoteke. Takšen zapis tako ne predstavlja pravega binarnega LZW izhoda, temveč zgolj tekstovno predstavitev kodnih mest, zaradi česar je njegova velikost bistveno večja od dejanske binarne kompresije.

4 Zaključek

Implementirali smo LZW dekodirnik in z MD5 preverjanjem potrdili pravilnost njegovega delovanja za vse obravnavane datoteke. Kompresija je pri besedilu najučinkovitejša (razmerje 3.4), dobro deluje tudi pri surovih slikah (1.8), medtem ko sta ZIP in MP3 zaradi že prisotne kompresije bistveno manj odzivna. Pri delnem kodiranju besedila učinkovitost narašča z velikostjo vhodnih podatkov, saj se slovar postopno izgradi. Teoretična velikost LZW odraža dejansko kompresijsko sposobnost algoritma, medtem ko je velikost tekstovne predstavitve kodnih mest odvisna od načina zapisa.

Literatura

[1] N. Pavešić, *Informacija in kodi*, 2. izdaja. Ljubljana: Založba FE in FRI, 2010.

[2] T. A. Welch, "A Technique for High-Performance Data Compression", *Computer*, 1984.

[3] P. Deutsch, “DEFLATE Compressed Data Format Specification”, *RFC 1951*, 1996.

[4] Python Documentation — `https://docs.python.org/3/`

[5] Y. Wu, S. Lonardi, and W. Szpankowski, “Error-resilient LZW data compression,” in *Proceedings of the Data Compression Conference (DCC’06)*, 2006, pp. 193–202. doi: 10.1109/DCC.2006.33.