Stiskanje podatkov

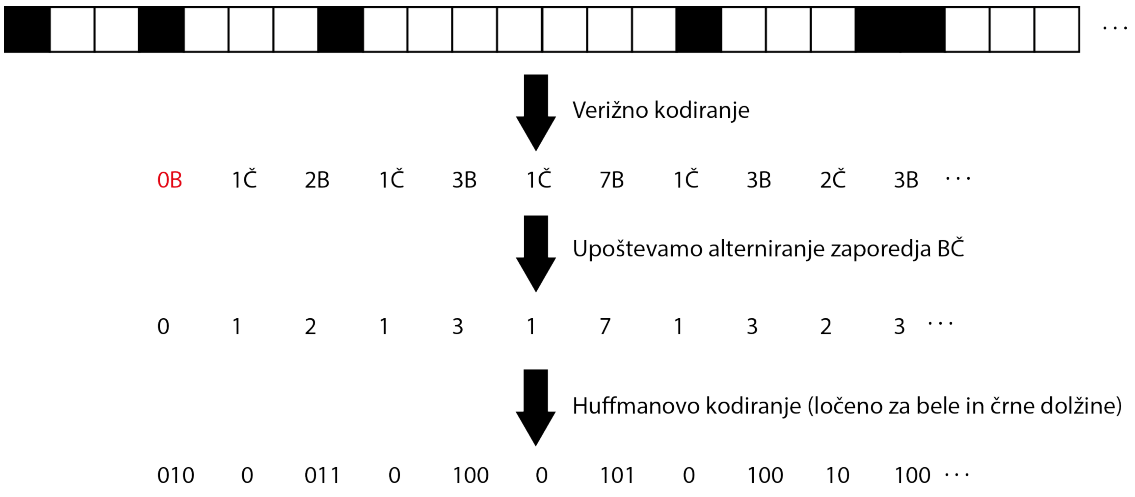
Uvod

Tokrat se bomo posvetili problemu brezizgubnega stiskanja podatkov [1]. Poskusili bomo implementirati nekoliko prirejeno obliko postopka za stiskanje podatkov, ki ga uporabljajo telefaksi[1] pri pošiljanju slik in dokumentov preko telefonskega omrežja [2]. Postopek temelji na kombinaciji verižnega kodiranja (ang. run-length encoding) in Huffmanovega kodiranja ter je zelo učinkovit pri kodiranju črno-belih slik.

Postopek stiskanja slik po standardu ITU-T T.4

V osnovi sliko, ki se pošilja po telefaksu, sestavlja poljubno število vrstic. V vsaki vrstici je 1728 črno-belih slikovnih točk. Kodiranje izvajamo nad vsako vrstico posebej. Najprej izvedemo verižno kodiranje, ki zaporedja belih ali črnih slikovnih točk nadomesti z njihovimi dolžinami. Standard predvideva, da se vsaka vrstica vedno začne z belo slikovno točko; če temu ni tako, na začetek vrstice umetno dodamo zaporedje belih slikovnih točk dolžine nič.

Sledi Huffmanovo kodiranje, ki se izvaja nad dolžinami zaporedij, ločeno za črne in za bele slikovne točke. Standard že vnaprej predpisuje Huffmanove kodne zamenjave za vse možne dolžine. Te kodne zamenjave so bile sestavljene na podlagi statistične analize večjega števila dokumentov. V naši nalogi teh predpisanih kodnih zamenjav ne bomo uporabili, ampak jih bomo izračunali na podlagi frekvenc dolžin zaporedij, ki se pojavijo v celotni sliki, ki jo želimo kodirati. Zgraditi moramo dve Huffmanovi drevesi: eno za zaporedja črnih dolžin in eno za zaporedja belih dolžin. Iz obeh dreves nato izluščimo dolžine kodnih zamenjav in uporabimo postopek za grajenje kanoničnih Huffmanovih kodnih zamenjav[2]. S temi nato nadomestimo dolžine zaporedij črnih in belih slikovnih točk v vsaki vrstici posebej.



Slika 1: Postopek kodiranja dela vrstice slike.

[1] `https://en.wikipedia.org/wiki/Fax`
[2] `http://en.wikipedia.org/wiki/Canonical_Huffman_code`

Huffmanov kod

Algoritem za kodiranje kot rezultat vrne optimalne dolžine kodnih zamenjav glede na podane verjetnosti znakov. V našem primeru znake predstavljajo dolžine zaporedij črnih ali belih slikovnih točk. Psevdo-koda:

```
1. Ustvari seznam osnovnih znakov L(1..n) s pripadajočimi verjetnostmi.
2. Ponavljaj dokler ne pridemo do sestavljenega znaka z verjetnostjo 1:
    - v seznamu L poišči znaka z najmanjšima verjetnostma[3];
    - ustvari nov sestavljen znak;
    - njegova verjetnost naj bo vsota verjetnosti obeh znakov;
    - najdena znaka označi kot uporabljena;
    - nov sestavljen znak dodaj v seznam L;
    - zabeleži si, iz katerih znakov je sestavljen.
3. Za vse sestavljene znake v L (od zadnjega proti prvemu):
    - znak razstavi, tj. poisci oba znaka, ki ga sestavljata;
    - obema znakoma priredi trenutno dolžino kodne zamenjave povečano za 1.
```

Dobili smo dolžine kodnih zamenjav za vse znake, ki nastopajo v vhodnem sporočilu. Sedaj želimo za vsak znak poiskati kodno zamenjavo. V tej nalogi je potrebno kodne zamenjave zapisati v kanonični obliki. Kanonična oblika omogoča zapis kodne tabele v zelo kompaktni obliki. Vse, kar potrebujemo za dekodiranje, je tabela znakov s pripadajočimi dolžinami kodnih zamenjav.
Psevdo-koda:

```
1. Uredi znake po dolžinah kodnih zamenjav (KZ)
   in nato še po njihovi vrednosti (naraščajoče).
2. KZ = zeros(1, dolžina KZ);
3. Dokler so znaki na vhodu:
    - izstavi znak in KZ;
    - KZ = (KZ + 1) << ((dolzina naslednje KZ) - (dolzina trenutne KZ));
```

Naloga

Napišite funkcijo z imenom `naloga2` v programskem jeziku Octave. Funkcija mora implementirati kodiranje črno-belih slik na osnovi zgoraj opisanega postopka. Vhodni parameter funkcije `vhod` je matrika ničel (črne slikovne točke) in enic (bele slikovne točke), ki predstavlja sliko, ki jo želimo poslati po telefaksu. Matrika vsebuje poljubno število vrstic, vsaka vrstica pa vsebuje 1728 slikovnih točk. Izhodni argumenti funkcije so štirje:

- zakodirana slika `izhod`, ki jo podate v obliki vrstičnega vektorja, kjer so vse vrstice zlepljene skupaj;

[3] V primeru, da je več znakov z enakimi verjetnostmi, upoštevajte njihovo vrednost - znak najprej izbiramo glede na stevilo elementov, ki jih vsebuje, nato glede na vrednost (nižje ima prednost).

- kompresijsko razmerje `R`, ki se izračuna kot $\frac{|\text{izhod}|}{|vhod|}$, kjer je $|x|$ število elementov vektorja oziroma matrike x;
- tabela dolžin kodnih zamenjav dolžin belih slikovnih točk `kodBela`[4];
- tabela dolžin kodnih zamenjav črnih slikovnih točk `kodCrna`[4].

Prototip funkcije:

```
function [izhod, R, kodBela, kodCrna] = naloga2(vhod)
% Izvedemo kodiranje vhodne binarne slike (matrike) vhod
% po modificiranem standardu ITU-T T.4.
% Slika vsebuje poljubno stevilo vrstic in 1728 stolpcev.
%
% vhod      - matrika, ki predstavlja sliko
% izhod     - binarni vrsticni vektor
% R         - kompresijsko razmerje
% kodBela   - matrika dolzin zaporedij belih slikovnih tock
%                     in dolzin kodnih zamenjav
% kodCrna   - matrika dolzin zaporedij crnih slikovnih tock
%                     in dolzin kodnih zamenjav


izhod = NaN;
R = NaN;
kodBela = [];
kodCrna = [];

end
```

[4]Matrika s toliko vrsticami kot je število različnih dolžin zaporedij in dvema stolpcema: v prvem se nahaja dolžina zaporedja, v drugem pa dolžina kodne zamenjave.

Testni primeri

Na učilnici se nahaja arhiv `naloga2.zip`, ki vsebuje tri testne slike, za katere imate podane vhod in izhode. Primeri so podani v obliki datotek .mat, ki jih naložite v Octave s pomočjo ukaza `load ime_datoteke.mat`. Priloženo imate tudi funkcijo `test_naloga2`, ki jo lahko uporabite za preverjanje pravilnosti rezultatov, ki jih vrača vaša funkcija (primer klica: `test_naloga2('primeri',1);`). Pri testiranju vaše funkcije upoštevajte naslednje omejitve:

- rezultat je pravilen:
  - če se od danega razlikuje za manj kot 10^{-6} v primeru `R`;
  - vektor `izhod` in se mora popolnoma ujemati z rešitvami, ni pa čisto nujno, če ste dvoumnosti v implementaciji reševali drugače kot mi. V primeru, da ni ujemanja, bomo med preverjanjem pognali dekodirnik nad `izhod`. Če se bo dekodirana slika ujemala z `vhod`, je rezultat pravilen.
- izvajanje funkcije je časovno omejeno na 120 sekund.

Namigi

Uporabne funkcije: `sortrows()`, `dec2bin()`, `find()`, `unique()`, `histc()`, `min()`.

Literatura

[1] D.G. Luenberger: Information Science, Princeton University, pogl. 4, 2006.

[2] Protocols for Telematic Services-Standardization of Group 3 facsimile apparatus for document transmission, ITU-T T.4, CCITT.