Laboratorijske vježbe iz Teorije informacije

Nenad Markuš

nenad.markus@fer.hr

1 Statistička analiza izvorišta informacije

Na službenoj stranici predmeta dostupni su programi koji predstavljaju diskretna izvorišta informacije: `i1.c`, `i2.c`, `i3.c` i `i4.c`. Sva generiraju simbole iz abecede $\{x, y\}$. Proučite njihov izvorni kôd i skicirajte ih kao 'konačne automate', kao u poglavlju 1.1 zbirke zadataka *Teorija informacije i kodiranje* (Ilić, Bažant, Beriša).

Pretvorite sva izvorišta u strojni kôd upotrebom C prevodioca. Pokretanjem dobivenih izvršnih datoteka na standardni se izlaz ispisuju generirani nizovi simbola. U svrhu daljnje analize, korisno je te nizove pohraniti u zasebne datoteke. To se jednostavno čini preusmjeravanjem standardnog izlaza programa u datoteku određenog imena. Na primjer, ako želimo generirati niz izvorištem 1 i pohraniti ga u datoteku `i1.txt`, u konzolu unosimo sljedeću naredbu:

```
i1.exe > i1.txt
```

Zadatak 1. Nacrtajte razdiobe vjerojatnosti simbola za sva izvorišta i izračunajte njihove entropije (preporučujemo da proračun i crtanje izvedete u Pythonu ili MATLAB-u). Uočavate li neke bitne razlike između njih?

Zadatak 2. Nacrtajte razdiobe pojavljivanja združenih 2, 4 i 8 simbola. Izračunajte entropije tih razdioba. Zapažate li sada značajnije razlike između izvorišta? Očekujete li da će neka od zadanih izvorišta generirati nizove koji će se moći bolje kompresirati?

Zadatak 3. Istražite ta očekivanja upotrebom stvarnog programa za kompresiju podataka (`http://en.wikipedia.org/wiki/Zip_(file_format)`). Numeričke vrijednosti unesite u tablicu oblika 1. Procijenite entropije svih izvorišta na temelju nje. Objasnite rezultate.

Izvorište	Veličina niza [kB]	
	Prije kompresije	Nakon kompresije
`i1`		
`i2`		
`i3`		
`i4`		

Tablica 1: Učinkovitost kompresije za zadana izvorišta.

2 Kompresija podataka

Kompresija podataka postupak je predstavljanja informacije u što manje memorijskog prostora i jasna je njena korisnost u velikom broju područja. U programskom jeziku C[1] potrebno je implementirati i usporediti dvije metode kompresije podataka, prema uputama u nastavku.

2.1 Huffmanova metoda

Potrebno je implmentirati Huffmanovu metodu kompresije podataka. Koder otvara ulaznu datoteku u binarnom načinu rada (`FILE* file = fopen("ulazna-datoteka", "rb");`) i pronalazi frekvenciju pojavljivanja pojedinih okteta (engl. *bytes*). Zatim, Huffmanovom metodom pridružuje binarni kôd svakom od njih i rezultat zapisuje u tekstualnu datoteku (prvi redak sadrži kôd pridružen oktetu `0x00`, drugi sadrži kôd pridružen `0x01` itd.). Napomena: potrebno je dodijeliti kôd i oktetima koji imaju frekvenciju pojavljivanja 0. Dalje, koder koristi generiranu tablicu pridruživanja za kompresiju ulazne datoteke. Rezultat zapisuje u izlaznu datoteku. Sav se izvorni kôd programa mora nalaziti u datoteci `huffkoder.c`. Prevođenjem dobiva se program koji preko komandne linije prima putanje do potrebnih datoteka, kako slijedi:

```
huffkoder.exe ulazna hufftablica.txt izlazna
```

Dekoder prima putanje do tekstualne datoteke koja sadrži tablicu pridruživanja binarnih kodova, kodirane datoteke i datoteke u koju će se zapisati rezultat dekodiranja. Sav se izvorni kôd mora nalaziti u datoteci `huffdekoder.c`. Primjer pozivanja:

```
huffdekoder.exe hufftablica.txt ulazna izlazna
```

Zadatak 4. Kolika je vremenska složenost dodjele kodova skupu od n simbola? Ako je $n = 256$, kolika je složenost kompresije datoteke od N okteta?

2.2 Metoda rječnika: LZW

Potrebno je implmentirati LZW metodu kompresije podataka. Koder čita ulaznu datoteku oktet po oktet, gradi rječnik i zapisuje izlazni niz simbola u izlaznu datoteku. Rječnik na početku sadrži samo ulaze simbole, tj.

$$D_i = i,\ i = 0, 1, 2, \ldots, 255$$

Maksimalna veličina neka mu je ograničena na 65536 simbola. Dakle, izlaz iz kodera su 16-bitovni brojevi (`unsigned short`, `uint16_t`). Učinkovito pretraživanje rječnika riješite prefiksnim stablom (`http://en.wikipedia.org/wiki/Trie`) ili nekom metodom

[1] Možete koristiti i C++, ali u tom slučaju ne smijete koristiti `std::map` i slične već gotove strukture podataka/algoritme koji bi olakšali izradu ovog labosa. Naime, poanta je studija, između ostaloga, naučiti studente samostalno implementirati takve strukture.

Izvorište	Veličina niza nakon kompresije [kB]	
	Huffman	LZW
`i1`		
`i2`		
`i3`		
`i4`		

Tablica 2: Učinkovitost kompresije za zadana izvorišta.

raspršenog adresiranja (engl. *hashing*, pogledati `http://eternallyconfuzzled.com/tuts/algorithms/jsw_tut_hashing.aspx` za praktične detalje). Sav se izvorni kôd kodera nalazi u datoteci `lzwkoder.c`. Primjer pozivanja:

```
lzwkoder.exe ulazna izlazna
```

Dekoder dekodira ulaznu datoteku i zapisuje rezultat u izlaznu datoteku. Sav se njegov izvorni kôd nalazi u datoteci `lzwdekoder.c`. Primjer pozivanja:

```
lzwdekoder.exe ulazna izlazna
```

Zadatak 5. Kolika je vremenska složenost LZW kompresije datoteke od N okteta?

2.3 Usporedba metoda

Zadatak 6. Popunite tablicu 2 i usporedite je s tablicom 1. Objasnite rezultate.

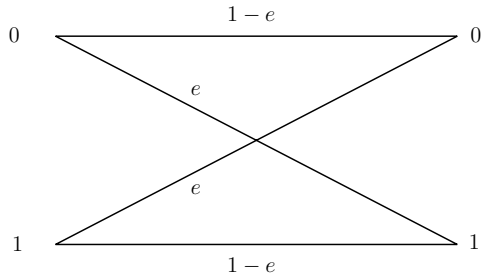
3 Zaštita podataka

U većini je stvarnih komunikacijskih sustava pogreška nepoželjna (ili čak nedopustiva). Stoga, stvarni sustavi koriste zaštitno kodiranje. Potrebno je implementirati model prijenosa podataka kanalom sa smetnjama.

3.1 Linearni binarni blok kôd $[16, 8, 5]$

Zadana je generirajuća matrica linearnog binarnog blok koda:

$$\mathbf{G} = \begin{bmatrix} 0001000100110011 \\ 0010001001011100 \\ 0100010011000011 \\ 1000100010101100 \\ 1100110111010111 \\ 1001101111101101 \\ 0011011110111011 \\ 0110111001111110 \end{bmatrix}$$



Slika 1: Binarni simetrični kanal s vjerojatnosti invertiranja bita e.

Svedite je na standardni oblik i odredite pripadnu matricu provjere pariteta.

Implementirajte koder i dekoder za gore opisani [16, 8, 5] kôd. Koder učitava ulaznu datoteku, štiti je zadanim kodom i proizvedeni niz bitova zapisuje u izlaznu datoteku. Sav se izvorni kôd nalazi u datoteci `linbinkoder.c`. Primjer pozivanja:

```
linbinkoder.exe ulazna izlazna
```

Dekoder provodi suprotan postupak, tj., dekodira ulaznu datoteku i rezultat zapisuje u izlaznu datoteku. Sav se izvorni kôd nalazi u datoteci `linbindekoder.c`. Primjer pozivanja:

```
linbindekoder.exe ulazna izlazna
```

3.2 Binarni simetrični kanal

Potrebno je napraviti program koji simulira binarni simetrični kanal (slika 1). Program učitava ulaznu datoteku u memoriju i svaki njezin bit invertira sa zadanom vjerojatnosti pogreške. Rezultat sprema u izlaznu datoteku. Izvorni kôd neka se nalazi u datoteci `binsimkanal.c`. Primjer pozivanja programa preko komandne linije:

```
binsimkanal.exe ulazna 0.01 izlazna
```

(invertira otprilike jedan posto bitova ulazne datoteke)

3.3 Prijenos podataka binarnim simetričnim kanalom

Na mrežnim stranicama predmeta dostupna je datoteka koja sadrži intenzitete piksela crno-bijele slike veličine 512×512. Sliku je moguće prikazati pomoću programa `prikazipiksele.c` na sljedeći način:

```
prikazipiksele.exe pikseli 512 512
```

Spomenutu je datoteku potrebno prenijeti implementiranim binarnim simetričnim kanalom za različite vjerojatnosti invertiranja pojedinog bita: $e = 0.1, 0.2, 0.3$. Prikažite sliku nakon prijenosa i opišite što uočavate.

Zaštitite datoteku implementiranim koderom.

Zadatak 7. Koliko se promijenila njezina veličina? Prenesite je binarnim simetričnim kanalom s ranije navedenim vjerojatnostima invertiranja bita. Dobivene datoteke dekodirajte i prikažite. Je li zaštitno kodiranje smanjilo štetan utjecaj kanala?