Multimedijalni računarski sistemi

3. Kompresija podataka

- Algoritmi zasnovani na rečnicima
- Aritmetička kompresija

https://web.microsoftstream.com/video/91df2f2d-0a47-4d9d-a3be-fdf77dee1f7c

2020

Sadržaj

- Kodovi zasnovani na rečnicima
  - LZ
  - LZW
- Aritmetička kompresija

Kodovi zasnovani na rečnicima

- Familija kodova bazirana na upotrebi rečnika (*dictionary based*)
- **Rečnik** –tabela stringova koji se koriste u procesu kodovanja (npr. Reč „lekcija“ se nalazi na strani X4 u liniji Y4 tj. ima kod (X4,Y4) a rečenica “Ovo je lekcija” se npr. može kodirati nizom kodova (X1,Y1)(X2,Y2)(X3,Y3)(X4,Y4)
- Predstavnik LZ (Lempel-Ziv) kod
- Doživeo mnoge modifikacije u cilju poboljšanja karakteristika (LZ77, LZ78, LZW, …)

Tehnike zasnovane na rečnicima

- Dve grupe tehnika:
  - **Statičke tehnike** – rečnik je generisan pre kodovanja i on se ne menja niti u procesu kodovanja niti u procesu dekodovanja
  - **Dinamičke tehnike** – rečnik se kreira dinamički (“u letu”) u toku samog procesa kodovanja na predajnoj strani (i ponekada i na prijemnoj strani)

LZ (Lempel-Ziv) kod

- Spada u kategoriju dinamičkih tehnika
- Predložili Abraham Lempel i Jacob Ziv, 1977 godine
- Postoje modifikacije ove tehnike korišćene za kompresiju bez gubitaka (LZ77, LZ78, …)
- Jedna modifikacija LZW (Lempel-Ziv-Welch) – 1984., Unix-ova *compress* komanda
- TIFF (Tag Image File Format) format se zasniva na LZ kodu

LZ (Lempel-Ziv) kod

- LZW korišćen u GIF i V.42 (komunikacija preko modema)
- Terry Welch patentirao deo svog algoritma i firma nosilac patentnih prava podnela veliki broj tužbi oko primene LZW u softverima koji su koristili GIF kompresiju što je rezultovalo u smanjenju primene LZW i povećanu upotrebu algoritama baziranih na LZ77
- DEFLATE algoritam - baziran na LZ77 i Huffmanovom algoritmu.
- DEFLATE u osnovi PNG, GZIP, ZIP
- LZMA (Lempel-Ziv-Markov) osnova 7-zip kodera

Princip LZ koda

- Deo poruke se menja referencom na identični deo koji se pojavio ranije u poruci.
- Podrazumeva se da je u opštem slučaju referenca kraća od dela poruke koju kodira
- Rečnik se dinamički kreira u toku obrade poruke

LZ kompresija - 1

abbababbbaabaa

Dužina ulaznog niza 14 karaktera

a	b	ba	bab	bb	aa	baa

Broj podnizova 7

LZ kompresija - 2

- String je podeljen u podstringove koji formiraju rečnik
- Sekvencijalni postupak
- U svakom koraku algoritam bira najkraći podstring koji počinje na tekućoj poziciji i ne nalazi se u rečniku tj. string $x=ys$ gde je y podstring koji se nalazi u rečniku a x se ne nalazi u rečniku i $s \in \Sigma$ (Σ je skup ulaznih simbola).

LZ kompresija - 3

- x=ys
- x se kodira uredjenom trojkom <p,l,s>, gde je p početna pozicija prethodnog pojavljivanja y, i l je dužina podstringa y.

- Algoritam se pomera za l+1 korak napred (duž ulaznog stringa) tako da sledeći podstring počinje od pozicije odmah iza x, i ubacuje x u rečnik.

LZ kompresija - 4

Primer:

x=ys

x → <p,l,s>

abbababbbaabaa

a	b	ba	bab	bb	aa	baa

Prvi simbol ima poziciju 1

<0,0,a>	<0,0,b>	<2,1,a>	<3,2,b>	<2,1,b>

<1,1,a>	<3,2,a>

LZ kompresija i sufiksno stablo

- Na svakom koraku se pronalazi najduži prefix koji se ranije već pojavio u ulaznom tekstu tj. nalazi se u rečniku.

- Ova operacija se efikasno može izvršiti primenom sufiksnog stabla.

LZ kompresija - prozor

- U praktičnim realizacijama rečnik nije proizvoljno dugačak i često je ograničen na razumnu meru.

- Često se koristi prozor fiksne dužine tj. traženje podstringa se ograničava na podstring koji se ranije već pojavio a ne prelazi okvir korišćenog prozora.

LZW (Lempel-Ziv-Welch) algoritam

- Statički načini za kodiranje zahtevaju prethodno znanje o podacima koji se kodiraju.
- Univerzalne šeme za kodiranje kao LZW ne zahtevaju nikakvo ranije poznavanje i generišu se u toku samog procesa tj "**u hodu**".
- LZW se primenjuje za kompresiju proizvoljnih podataka zbog svoje jednostavnosti i prilagodljivosti.
- LZW se nalazi u osnovi mnogih alata za povećanje kapaciteta HDa
- LZW kompresija koristi kodnu tabelu sa 4096 ulaza tj. elemenata u tabeli.

LZW algoritam - 2

- Kodovi od 0-255 u kodnoj tabeli su uvek dodeljeni za predstavljanje simbola iz ulaznog fajla.
- Kada počinje proces kodiranja tabela u startu ima samo prvih 256 elemenata dok su ostali elementi prazni.
- Ostali elementi tabele od 256 do 4095 predstavljaju podnizove u okviru ulaznog fajla.
- U toku procesa kodiranja identifikuju se podstringovi koji se ponavljaju i dodaju se u kodnu tabelu.
- Proces dekodiranja se svodi na uzimanje svakog koda iz kompresovanog fajla i transliranje u originalni podstring korišćenjem izgenerisane kodne tabele.

LZW algoritam - kodiranje

```
1     Inicijalizuj kodnu tabelu pojedinačnim karakterima stringa
2     P = prvi ulazni karakter
3     WHILE not end of ulaznog strima
4           C = sledeći ulazni karakter
5           IF se P + C nalazi u tabeli podstringova
6             P = P + C
7           ELSE
8             output kod za P
9             dodaj P + C u tabelu podstringova
10            P = C
11    END WHILE

12    output kod za P
```

LZW algoritam - Primer

Primer:

Korišćenjem LZW algoritma izvršiti kompresiju ulaznog stringa

B A B A A B A A A

LZW algoritam - Primer

BABAABAAA

P=A

ENCODER	OUTPUT	STRING	TABLE
output code	representing	codeword	string
66	B	256	BA

LZW algoritam - Primer

BABAABAAA P=B

ENCODER OUTPUT		STRING TABLE	
output code	representing	codeword	string
66	B	256	BA
65	A	257	AB

LZW algoritam - Primer

BABAABAAA

P=A

ENCODER OUTPUT		STRING TABLE	
output code	**representing**	**codeword**	**string**
66	**B**	**256**	**BA**
65	**A**	**257**	**AB**
256	**BA**	**258**	**BAA**

LZW algoritam - Primer

BABAABAAA P=A

ENCODER OUTPUT		STRING TABLE	
output code	**representing**	**codeword**	**string**
66	**B**	**256**	**BA**
65	**A**	**257**	**AB**
256	**BA**	**258**	**BAA**
257	**AB**	**259**	**ABA**

LZW algoritam - Primer

BABAABAAA P=A

ENCODER OUTPUT		STRING TABLE	
output code	**representing**	**codeword**	**string**
66	**B**	**256**	**BA**
65	**A**	**257**	**AB**
256	**BA**	**258**	**BAA**
257	**AB**	**259**	**ABA**
65	**A**	**260**	**AA**

LZW algoritam - Primer

BABAABAAA P=AA

ENCODER OUTPUT		STRING TABLE	
output code	**representing**	**codeword**	**string**
66	**B**	**256**	**BA**
65	**A**	**257**	**AB**
256	**BA**	**258**	**BAA**
257	**AB**	**259**	**ABA**
65	**A**	**260**	**AA**
260	**AA**		

LZW algoritam - dekompresija

- **LZW dekompresor kreira istu tabelu podstringova u toku procesa dekompresije tj. tabela se ne prenosi.**
- **Polazi se sa tabelom koja inicijalno ima 256 elemenata pridruženih pojedinačnim karakterima.**
- **Tabela podstringova se dopunjuje za svaki karakter u ulaznom strimu izuzev za prvi.**
- **Dekodiranje se svodi na čitanje kodova i njihovo transliranje u podstringove na osnovu izgradjene tabele podstringova.**

LZW algoritam - dekompresija

```
1    Inicijalizacija tabele podstringova pojedinačnim karakterima
2    OLD = prvi ulazni kod
3    output translacije za OLD (tj S(OLD) )
4    WHILE not end of ulazni strim
5        NEW = sledeći ulazni kod
6        IF se NEW ne nalazi u tabeli podstringova
7                S = translacija od OLD (tj S(OLD) )
8                S = S + C (prvi karakter od S)
9                Output S
10               Upiši S u tabelu podstringova
9        ELSE
10               S = translacija od NEW
11               output S
12               C = prvi karakter od S
13               S(OLD) + C upiši u tabelu podstringova
14       OLD = NEW
15   END WHILE
```

LZW dekompresija - primer

Primer:

Izvršiti dekompresiju ulaznog niza korišćenjem LZW dekompresora

<66><65><256><257><65><260>.

LZW dekompresija - primer

<66><65><256><257><65><260>

Old = 66 S = B

New = 65 C = A

ENCODER OUTPUT	STRING TABLE	
string	codeword	string
B		
A	256	BA

LZW dekompresija - primer

<66><65><256><257><65><260>

Old = 256 S = BA

New = 256 C = B

ENCODER OUTPUT	STRING TABLE	
string	**codeword**	**string**
B		
A	**256**	**BA**
BA	**257**	**AB**

LZW dekompresija - primer

<66><65><256><257><65><260>

Old = 257 S = AB

New = 257 C = A

ENCODER OUTPUT	STRING TABLE	
string	codeword	string
B		
A	256	BA
BA	257	AB
AB	258	BAA

LZW dekompresija - primer

<66><65><256><257><65><260>

Old = 65 S = A

New = 65 C = A

ENCODER OUTPUT	STRING TABLE	
string	**codeword**	**string**
B		
A	**256**	**BA**
BA	**257**	**AB**
AB	**258**	**BAA**
A	**259**	**ABA**

LZW dekompresija - primer

<66><65><256><257><65><260>

Old = 260 S = AA

New = 260 C = A

ENCODER OUTPUT	STRING TABLE	
string	codeword	string
B		
A	256	BA
BA	257	AB
AB	258	BAA
A	259	ABA
AA	260	AA

LZW - karakteristike

- LZW dobro kompresuje sekvence karaktera koje se često ponavljaju.
- Složenost i kodiranja i dekodiranja je O(N) tj O(M) gde je N dužina ulaznog niza a M dužina koda

- Za kodne reči se koristi **12 bita** pa svaki karakter kodiran samostalno ne smanjuje veličinu kompresovanog niza već je povećava.

- U primeru za reč **BABAABAAA** koja zauzima 72 bita (9x8 =72) dobija se kod (**<66><65><256><257><65><260>**) koji takodje zauzima 72 bita tj 6 x 12 = 72. Medjutim u realnim primerima se značajno dobija na kompresiji.

- Prednosti LZW u odnosu na Huffmanov algoritam:
  - LZW ne zahteva prethodnu informaciju o ulaznom nizu.
  - LZW može da izvrši kompresiju u jednom prolazu.
  - LZW je jednostavan i samim tim dosta brz algoritam.

Ograničenja LZW

- Šta se dešava kada rečnik postane suviše velik (tj. kada svih 4096 lokacija bude iskorišćeno)?
- Neke opcije koje se standardno koriste:
  - Ne dodavati nove elemente i koristiti postojeće.
  - Odbaciti rečnik kada on dostigne odgovarajuću veličinu.
  - Odbaciti rečnik kada on više nije efikasan za kompresiju.
  - Obrisati elemente od 256-4095 i početi sa izgradnjom rečnika ispočetka.
  - Neke inteligentne šeme grade tabelu podstringova na osnovu poslednjih N ulaza.

LZW vs Huffman

Veličina kompresovanog fajla u odnosu na početnu veličinu

Tip fajla	Kodovano sa Huffman	Kodovano sa LZW
C sors kod	65%	45%
Mašinski kod	80%	55%
Textualni fajl	50%	30%

Problemi Huffmanovog kodiranja

- Sa stanovišta teorije informacija Huffmanov kod nije sasvim optimalan.
- Kodne reči moraju uvek da imaju ceo broj bitova tako da ne odgovaraju u potpunosti frekvenci pojavljivanja.
- Ako je za neki simbol p=0,9, optimalan broj bita za predstavljanje je 0,15. On bi se Huffmanovim kodom kodirao jednim bitom što je 7 puta veće od teorijskog optimuma

Problemi Huffmanovog kodiranja

- Kod malih alfabeta Huffmanov kod ne daje dobre rezultate. Npr. slika sa dve boje. Dovoljno je 1 bit po pixelu pa Huffmanovo kodiranje ne može da ostvari kompresiju. Alternativa je da se pixeli slike grupišu pa da se Huffmanov kod primeni na grupe pixela
- Ove probleme rešava aritmetička kompresija tj. kodovanje
- Ideja-celu poruku predstaviti realnim brojem iz intervala [0,1)

Aritmetička kompresija

- Interval (0, 1) se sukcesivno deli na podintervale u skladu sa frekvencom pojavljivanja sledećeg simbola.
- Svaki novi podinterval predstavlja jedan simbol.
- Po završetku procesa najmanji realan broj (binarno predstavljen) sadržan u odredišnom intervalu se bira kao kod date poruke

Aritmetička kompresija - kodiranje

- Počinje se sa prvim karakterom ulaznog niza sa celokupnim intervalom (0-1)
- Čita se sledeći karakter i deli interval u skladu sa frekvencijom pojavljivanja svih karaktera u alfabetu. Selektuje se interval koji odgovara tekućem karakteru.
- Ako se dodje do kraja ulaznog niza ili end simbola ide se na sledeći korak, u suprotnom se ide na prethodni korak
- Iz konačnog tekućeg intervala se selektuje realan broj koji se može predstaviti u računaru sa najmanjim brojem bitova. Ovaj broj je kod ulaznog stringa.

Aritmetička kompresija-dekodiranje

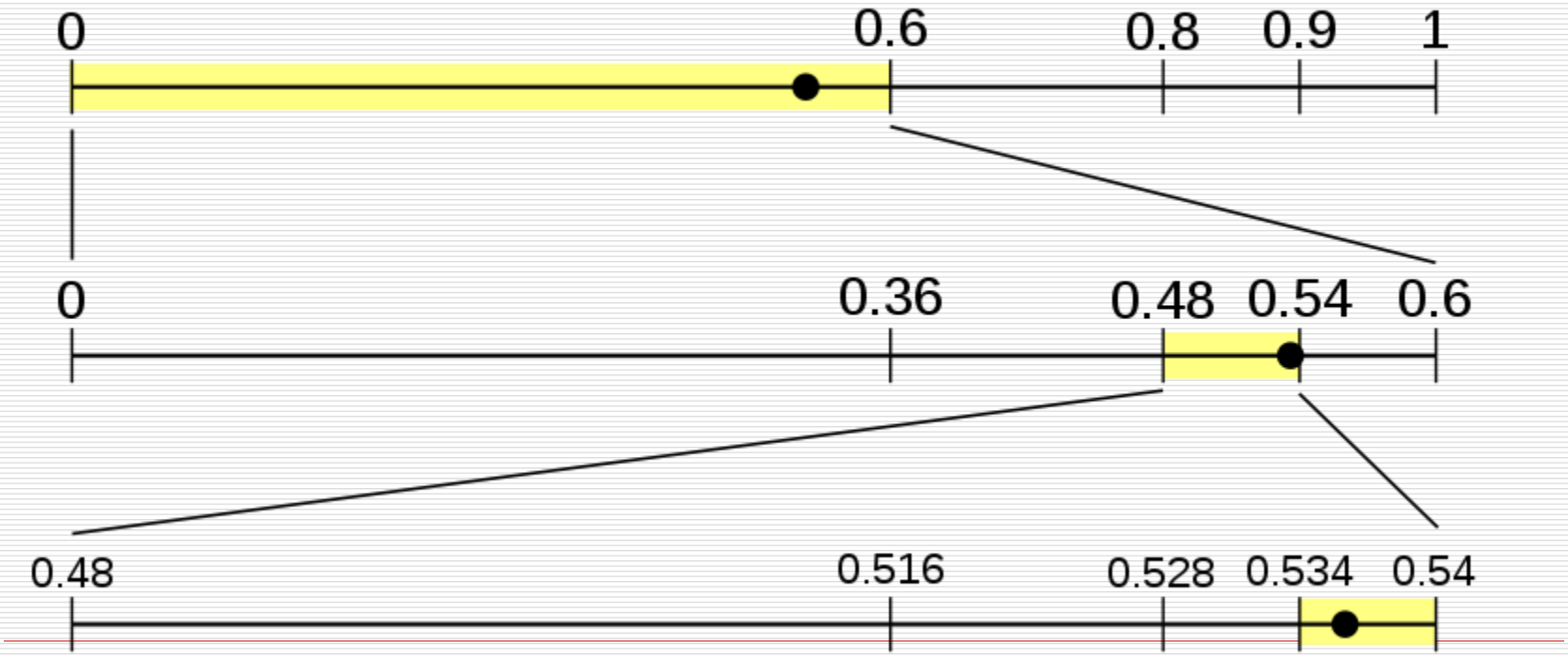
- Podeli se interval (0,1) u skladu sa frekvencijom pojavljivanja simbola kao što je opisano u algoritmu za kodiranje do maksimalne veličine jednake dužini poruke
- Realan broj koji predstavlja kodiranu poruku jedinstveno identifikuje jedan podinterval.
- Ovaj podinterval jedinstveno predstavlja polaznu poruku.

Aritmetička kompresija - primer

- Alfabet=(A,B,C)
- Frekvencije pojavljivanja:A(0,2), B(0,3) i C(0,5)
- Poruke: ACB AAB (max dužina poruke je 3)
- Kodiranje prvog bloka ACB (3 podintervala u odnosu 2 : 3 : 5)
  - A (0- 0,2 - 0,5 - 1)
  - C (0- 0,04 – 0,1 – 0,2)
  - B (0,1 – 0,12 – 0,15 – 0,2)
  - Konačni interval je (0,12 – 0,15) biramo npr. 0,125 (predstavlja se kao 1/8 tj. 3 bita)

Dekodovanje 0.538

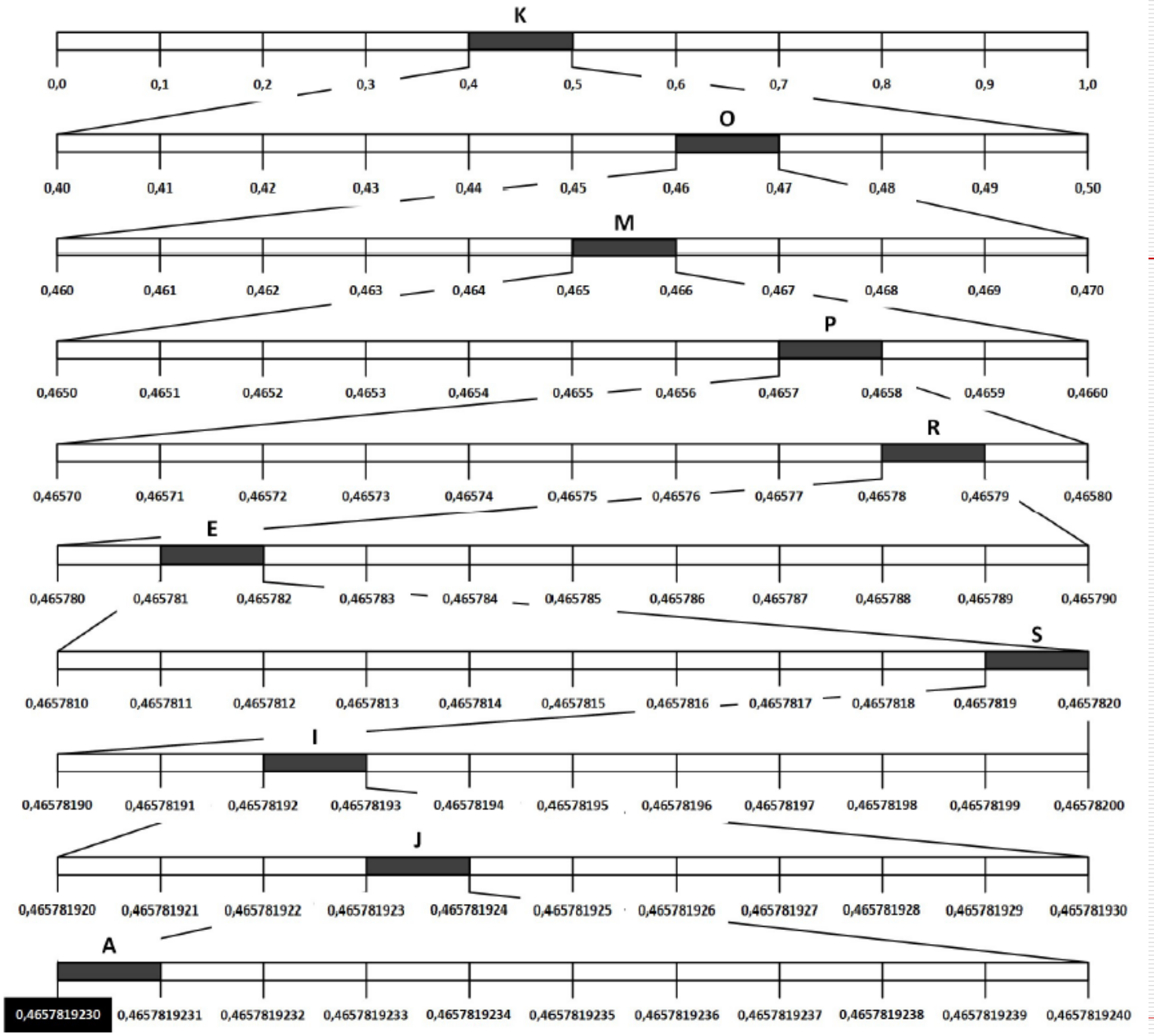
- p1=0.6; p2=0.2; p3=0.1; p4=0.1



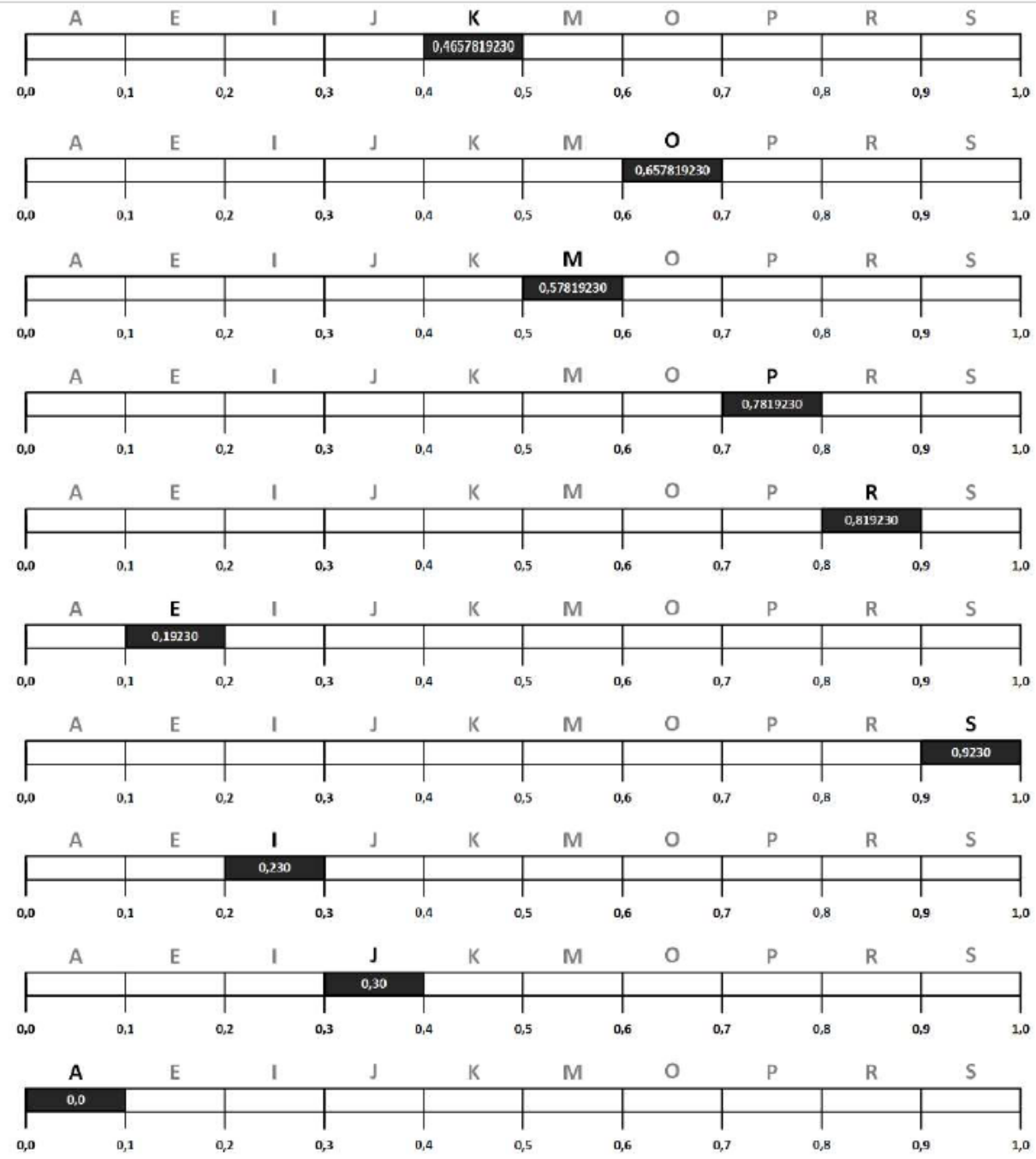
Primer:

- Podjednake verovatnoće (10 simbola)
- Ulaz: KOMPRESIJA

Znaci	Verovatnoća pojavljivanja	Interval
A	1/10	[0 , 0,1)
E	1/10	[0,1 , 0,2)
I	1/10	[0,2 , 0,3)
J	1/10	[0,3 , 0,4)
K	1/10	[0,4 , 0,5)
M	1/10	[0,5 , 0,6)
O	1/10	[0,6 , 0,7)
P	1/10	[0,7 , 0,8)
R	1/10	[0,8 , 0,9)
S	1/10	[0,9 , 0,1)



- 0,4657819230



- 0,4657819230
- U K int (0.4 – 0.5)
- 0,4657819230 – 0,4
- ~~0,0657819230 : 0,1~~
- 0,657819230
- U O int (0.6 – 0.7)
- 0,657819230 – 0,6
- 0,057819230 : 0,1
- 0,57819230
- …

Aritmetička kompresija

- Radi se sa realnim brojevima sa teorijski neograničenom tačnošću
- Šta se dešava kada kodiramo više megabajta podataka?
- Ako se karakter A pojavljuje sa verovatnoćom od 20% broj cifara potrebnih za kodiranje A vrlo brzo raste tj. (A<0,2; <0,04; <0,008; <0,0016 itd. (20% od prethodnog)
- Čak i za 32 i 64-bitne CPU registre samo par znakova može da se kodira
- U C progr. jeziku korišćenjem long double tipa maksimalno oko 18 znakova može da se kodira.

Rešenje

- Vodeće nule preneti i dobiti novi kod
- Npr. interval (0,00000000, 0,00110011)
- Dve nule se izbacuju
  - (0.000000??, 0.110011??)
  - Dopuna donje granice nulama i gornje jedinicama tj. (0.00000000, 0.11001111)
- Terminalni simbol završava proces kodiranja
- Inkrementalno aritmetičko kodiranje – korišćenje celobrojne aritmetike
- **Problem**: jedan pogrešan bit narušava celu poruku!!!

Aritmetička kompresija-patenti

- Some US patents relating to arithmetic coding are listed below.
  - U.S. Patent 4,122,440 — (IBM) Filed 4 March 1977, Granted 24 October 1978 (Now expired)
  - U.S. Patent 4,286,256 — (IBM) Granted 25 August 1981 (Now expired)
  - U.S. Patent 4,467,317 — (IBM) Granted 21 August 1984 (Now expired)
  - U.S. Patent 4,652,856 — (IBM) Granted 4 February 1986 (Now expired)
  - U.S. Patent 4,891,643 — (IBM) Filed 15 September 1986, granted 2 January 1990 (Now expired)
  - U.S. Patent 4,905,297 — (IBM) Filed 18 November 1988, granted 27 February 1990 (Now expired)
  - U.S. Patent 4,933,883 — (IBM) Filed 3 May 1988, granted 12 June 1990 (Now expired)
  - U.S. Patent 4,935,882 — (IBM) Filed 20 July 1988, granted 19 June 1990 (Now expired)
  - U.S. Patent 4,989,000 — Filed 19 June 1989, granted 29 January 1991 (Now expired)
  - U.S. Patent 5,099,440 — (IBM) Filed 5 January 1990, granted 24 March 1992 (Now expired)
  - U.S. Patent 5,272,478 — (Ricoh) Filed 17 August 1992, granted 21 December 1993 (Now expired)
- Note: This list is not exhaustive. See the following link for a list of more patents.[4] The Dirac codec uses arithmetic coding and is not patent pending.[5]

Poređenje algoritama

	Aritmetička	**Character Huffman**	**Word Huffman**	**LZ**
Stepen kompresije	Vrlo dobar	Loš	Vrlo dobar	Dobar
Brzina kompresije	Sporo	Brzo	Brzo	Vrlo brzo
Brzina dekompresije	Sporo	Brzo	Vrlo brzo	Vrlo brzo
Potrebna memorija	Mala	Mala	Velika	Osrednja
Pattern matching	Ne	Da	Da	Da
Random Access	Ne	Da	Da	Ne

Huffman word– engleske reči

Alice was not a bit hurt, and she jumped up on to her feet in a moment: she looked up, but it was all dark overhead; before her was another long passage, and the White Rabbit was still in sight, hurrying down it. There was not a moment to be lost: away went Alice like the wind, and was just in time to hear it say, as it turned a corner, 'Oh my ears and whiskers, how late it's getting!' She was close behind it when she turned the corner, but the Rabbit was no longer to be seen: she found herself in a long, low hall, which was lit up by a row of lamps hanging from the roof.

xlAeRgEpfLP, JYrNBKAXOAGXQdRXv: YCOHr, AchlAMATWSAjFLpFNW; HHAGlHAEVACJ, JDGmEhlTXdaQ, pfLLAAIKi BRAK. EtlAeRGvOAYART: HjBPxBSDAmN, JlCQdCKOQQhHv, shHsRADz, 'jxBpADlJANT, CVARiAK'AHiFPPIKi!' EAlTnTohBVYHsDADz, AcDEhlBTAsOOAYAPn: YELCedRAsQ, gzZp, CJlVIAFBKBURApgUVNtkAHASGCSDL SSAI.

Huffman word kodna tabela

AA “y”
ABA “too”
ABB “tongue”
ABC “tm”
ABD “tiny ”
...
AF “t ”
AG “her ”
AH “s ”
AI “f”
AJ “had ”
AK “it”
AL “Alice”
AM “all ”

...
Azx “Knave”
Azy “Keep ”
Azz “J”
BA “S”
BB “out ”
BC “he ”
BDA “Its ”
BDB “IT ”
BDC “IT”
BDD “Hush”
BDE “Hart”
...
BI “what ”

BJ “you”
BK “up ”
BL “Project ”
BM “were ”
BN “if ”
BO “one ”
BP “went ”
BQ “have ”
BR “down ”
BS “like ”
BT “no ”
...
I “i”
J “and ”

K “n”
L “r”
M “s”
N “a”
O “to ”
P “t”
...
uy “tree ”
uz “tree”
v “I”
w “that ”
x “Alice ”
y “at ”
z “with ”

Pitanja

- ?????