Grafika Komputerowa i Multimedia

Wykład 10
Algorytmy kodowania arytmetycznego

Damian Grela
e-mail: **dgrela@pk.edu.pl**
http://www.dgrela.pl

Wprowadzenie

- Podobnie jak kodowanie **Huffmana**, kodowanie arytmetyczne bazuje na rozkładzie częstości (lub prawdopodobieństwa) występowania poszczególnych znaków alfabetu w kompresowanej wiadomości.
- Co więcej, udowodniono matematycznie, że **kod Huffmana jest szczególnym przypadkiem kodowania arytmetycznego.**

- W kodowaniu arytmetycznym zastosowanie znajdują te same metody określania rozkładu prawdopodobieństwa, co przy kodowaniu Huffmana, a więc rozkład statyczny (dany z góry, lub liczony dla całej wiadomości), dynamiczny (dla kolejnych bloków), lub adaptacyjny.

- Dodatkowo istnieje możliwość stosowania rozkładów warunkowych, na przykład możemy wykorzystać fakt, że prawdopodobieństwo wystąpienia „B” po „A” jest inne niż po innych znakach.
- Koder arytmetyczny pozwala na zmianę rozkładu częstości nawet co 1 znak, warunek jest taki, że dekoder musi znać te wszystkie rozkłady i zmieniać je tak samo jak koder.

Idea kodowania arytmetycznego

- Zupełnie inaczej wyglądają dalsze czynności. Nie jest tworzony słownik symboli.
- Cała wiadomość jest kodowana jako jedna długa liczba z przedziału <0, 1>.
- Na początku na bazie rozkładu częstości tworzymy zestaw przedziałów liczbowych. Każdy znak alfabetu dostaje swój podprzedział, którego szerokość jest proporcjonalna do częstości jego występowania w wiadomości.

Idea kodowania arytmetycznego

- Oczywiście suma szerokości przedziałów wynosi 1.
- Na rysunkach obok pokazano rozkład częstości pikseli w przykładowej linii znanej z poprzednich wykładów, oraz skonstruowane przedziały sumujące się do przedziału <0, 1>.

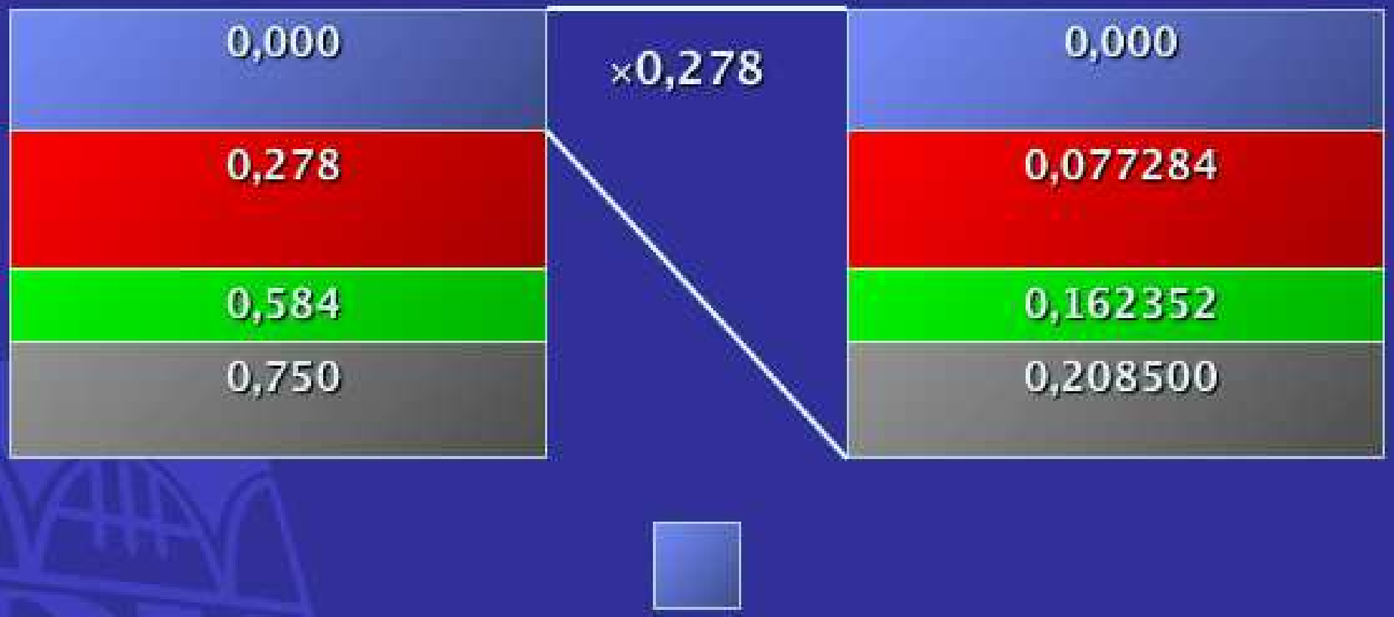
	10/36 = 0,278
	11/36 = 0,306
	6/36 = 0,166
	9/36 = 0,250

Idea kodowania arytmetycznego

- Kolor pierwszego piksela w linii wybiera nam podprzedział.
- Następnie cały zestaw przedziałów skalujemy tak, aby zmieścił się w dopiero co wybranym podprzedziale i znów wybieramy podprzedział zgodny z następnym elementem wiadomości.
- Proces ten dla kilku pierwszych pikseli linii pokazano na następnej stronie.

Idea kodowania arytmetycznego

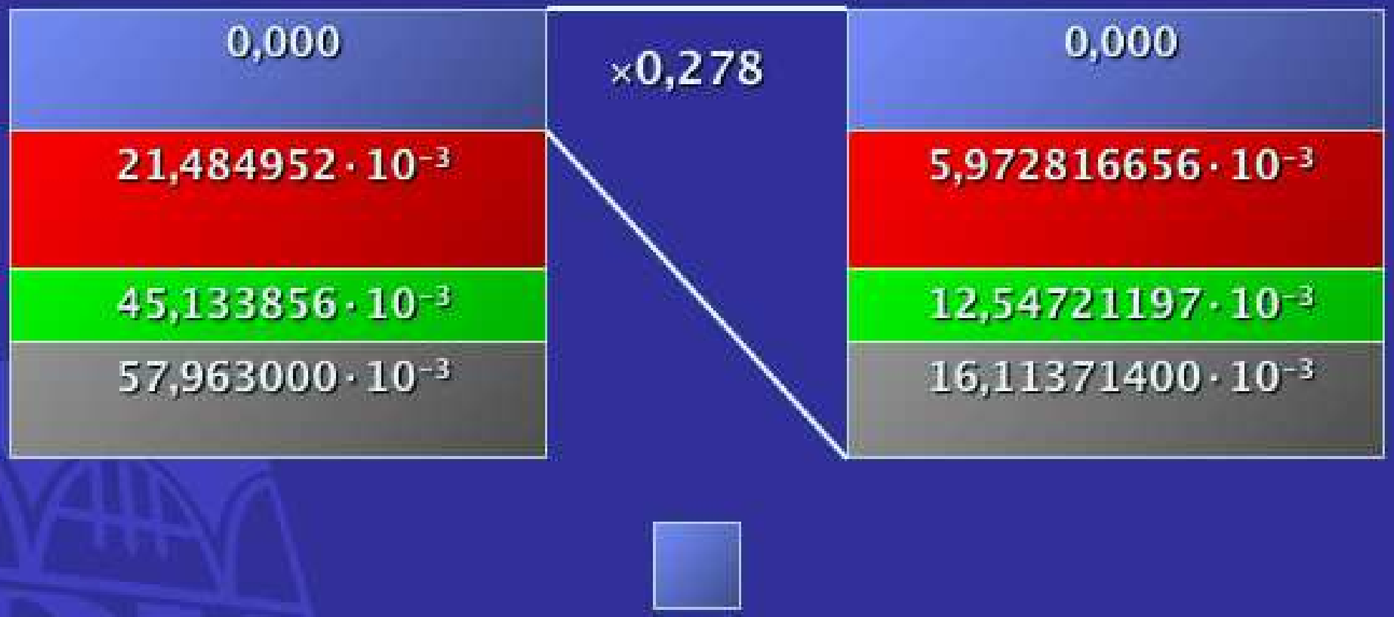
Idea kodowania arytmetycznego



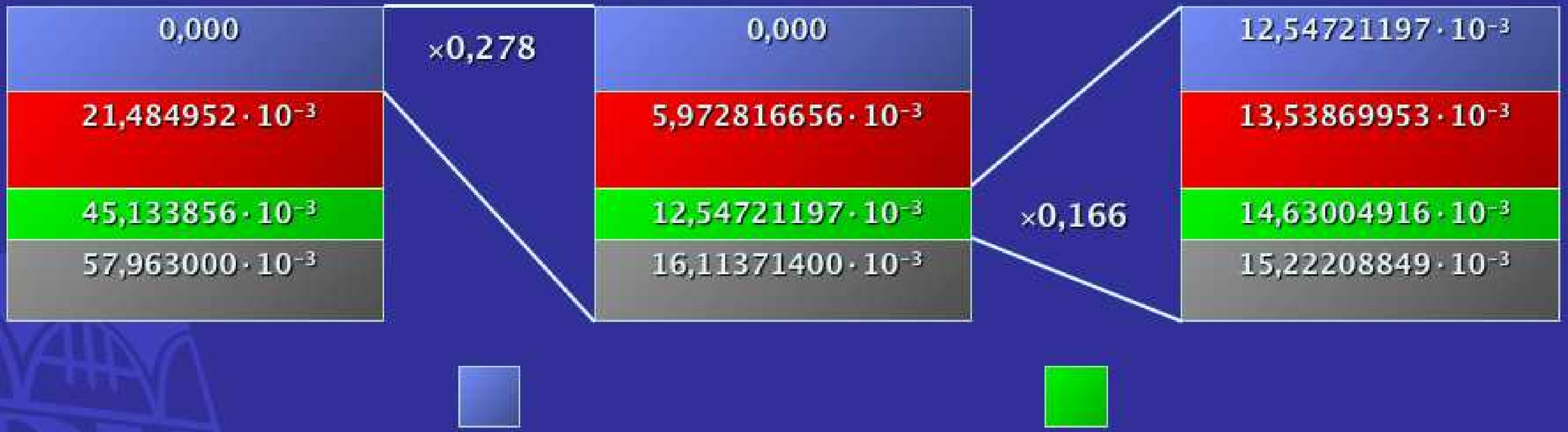
Idea kodowania arytmetycznego

Idea kodowania arytmetycznego

Idea kodowania arytmetycznego



Idea kodowania arytmetycznego



Idea kodowania arytmetycznego

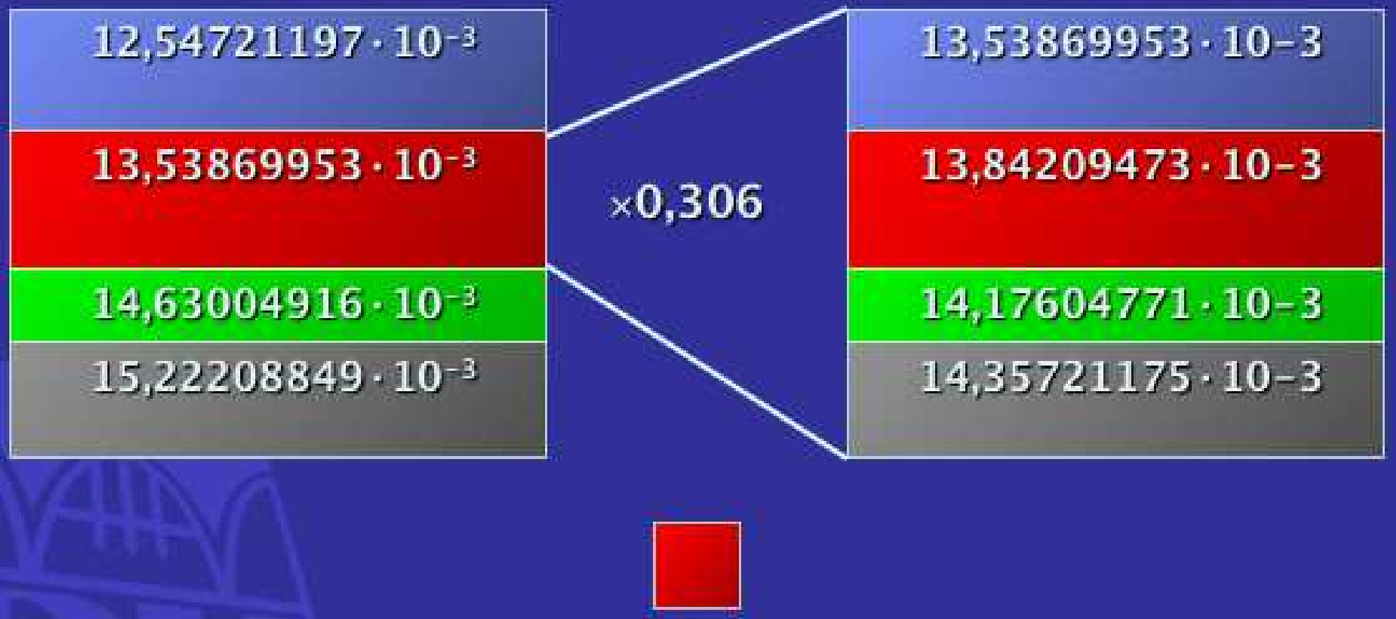
$12{,}54721197 \cdot 10^{-3}$

$13{,}53869953 \cdot 10^{-3}$

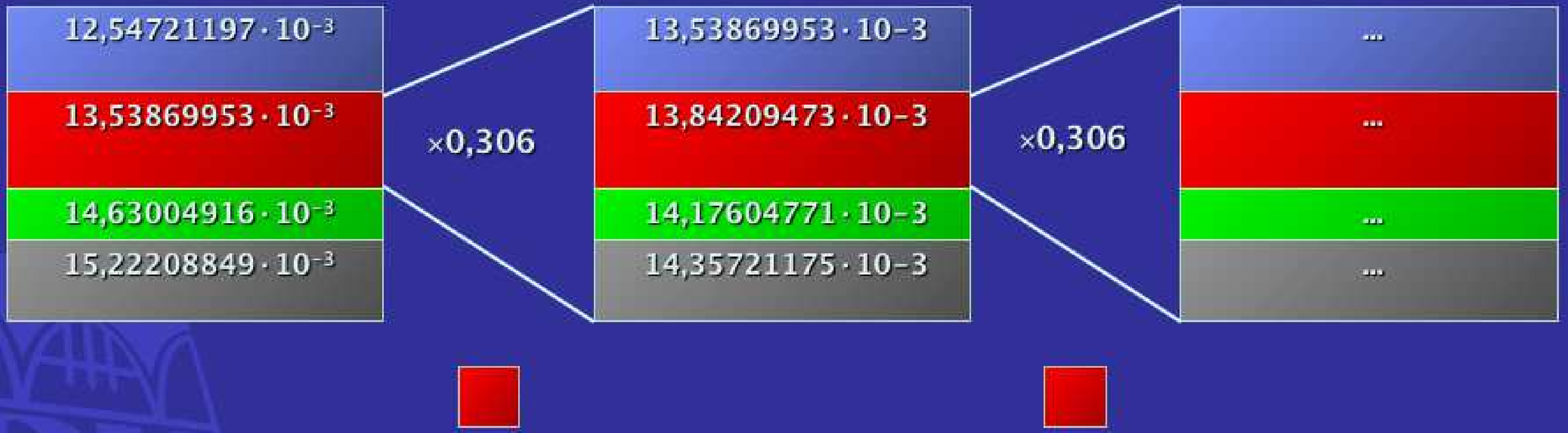
$14{,}63004916 \cdot 10^{-3}$

$15{,}22208849 \cdot 10^{-3}$

Idea kodowania arytmetycznego



Idea kodowania arytmetycznego



Renormalizacja przedziału

- Jest sprawą oczywistą, że liczba będąca wynikiem działania kodera artymetycznego staje się coraz dłuższa w miarę kompresowania kolejnych znaków wiadomości.
- To oczywiste, bo każdy kolejny przedział musi być zapisany z większą dokładnością (więcej cyfr po przecinku).
- Trzymanie całej liczby w pamięci mija się z celem, dlatego wykorzystuje się renormalizację przedziału.

Renormalizacja przedziału

- Renormalizacja polega w uproszczeniu na odrzuceniu najbardziej znaczących cyfr liczby, takich samych dla początku i końca przedziału. Cyfry te zostają zapisane do strumienia wyjściowego a następnie odrzucone. Dzięki temu koder pracuje na liczbach o skończonej precyzji.

Renormalizacja przedziału

- Po nadejściu zielonego piksela w przykładzie z poprzednich stron, przedział kodera wyniesie:

$$0,01417604771 - 0,01435721175$$

- Początkowe cyfry „0,014” są identyczne, a więc można je zapisać do strumienia, odrzucić i odjąć od przedziału otrzymując:

$$0,00017604771 - 0,00035721175$$

- Następnie renormalizujemy przedział o czynnik 1000, otrzymując:

$$0,17604771 - 0,35721175$$

Renormalizacja przedziału

- Należy oczywiście pamiętać, że praktyczne implementacje pracują na liczbach w systemie binarnym, dzięki czemu renormalizacja może być realizowana jako przesunięcia bitowe.
- Najczęściej renormalizacja odbywa się skokami co 1 bajt (8 cyfr binarnych).

Wstęp do kodowania Golomba

- Jak wspominane było ostatnio, postać optymalnego kodu Huffmana (1 symbol – 1 kod) zależy od rozkładu częstości (lub prawdopodobieństwa) występowania poszczególnych symboli w wiadomości.
- Jeżeli rozkład jest znany z góry, kod może zostać zdekodowany bez potrzeby przesyłania do dekodera tego rozkładu.
- Kody unarny i binarny skrócony są właśnie specjalizowanymi kodami dla ściśle określonych rozkładów.

Kod unarny

- Kod ten jest optymalny dla rozkładów silnie nierównomiernych o charakterze zbliżonym do geometrycznego (rozkład ściśle geometryczny to taki, że
  $P(x_i) = \alpha P(x_{i+1})$
  a więc częstość występowania każdego kolejnego elementu alfabetu jest α razy mniejsza od poprzedniego.

Kod unarny

- Warunkiem optymalności kodu unarnego jest $P(x_i) \geq P(x_{i+1}) + P(x_{i+2})$, a więc jeżeli posortujemy elementy według malejącej częstości występowania, to częstość danego elementu musi być większa lub równa sumie częstości dwóch następnych elementów.
- Warunek ten spełniają wszystkie rozkłady ściśle geometryczne o α większym lub równym tzw. liczbie złotego podziału równej około 1,618033989.

Kod unarny

- Warunek ten jest również spełniony przez wiele rozkładów zbliżonych do geometrycznych.
- Kod unarny jest bardzo prosty.
- Pierwszy element alfabetu, o największej częstości występowania otrzymuje kod **0**, zero jest w tym przypadku tzw. bitem stopu.

Kod unarny

- Kolejne elementy otrzymują kody złożone z odpowiedniej ilości jedynek i zerowego bitu stopu na końcu, a więc: **10**, **110**, **1110**, **11110**, **111110**, i tak dalej.
- Oczywiście możliwy jest kod odwrotny, w którym bitem stopu byłoby 1.
- Kod unarny (jak każdy szczególny przypadek kodu Huffmana) jest prefiksowy, co umożliwia jego jednoznaczne zdekodowanie.

Kod prefiksowy

- Kod prefiksowy lub przedrostkowy (ang. prefix code) – kod, w którym żadne ze słów kodowych nie jest przedrostkiem innego słowa;
  taki kod jest jednoznacznie dekodowalny.
- Dodatkowo każdy kod prefiksowy można reprezentować w formie drzew (dla kodów dwójkowych drzewo binarne).

Kod prefiksowy

- Dzięki tej cesze kody są jednoznacznie identyfikowane, nie ma potrzeby wstawiania dodatkowych informacji np. o tym, gdzie kończy się słowo kodowe (jest to jednoznaczne) albo jaką ma długość (długość każdego słowa kodowego jest znana z góry). Stosując kody prefiksowe, można uzyskać maksymalny stopień upakowania danych w różnych metodach kompresji.

Kod prefiksowy

- Dla przykładu mając kod nie będący prefiksowym:
  literze 'a' odpowiada bit 0, literze 'b' odpowiada bit 1,
  literze 'c' dwa bity 01 – kod litery 'a' jest prefiksem kodu litery 'c'.
  Przy takim przyporządkowaniu nie można jednoznacznie stwierdzić,
  co oznacza np. komunikat 0110 – może to być zarówno 'cba', jak i
  'abba'.

- Zmieniając kod na prefiksowy: 'a' – 0, 'b' – 10, 'c' – 11, ten sam
  komunikat ma jednoznaczną interpretację, tj. 'aca'.

Kod binarny skrócony

- Kod binarny skrócony stosowany jest do równomiernego rozkładu częstości znaków alfabetu, ale w przypadku, gdy ilość znaków alfabetu nie jest równa potędze dwójki (gdy jest równa, optymalnym kodem jest zwykłe kodowanie binarne, co prawda nie jest to kodowanie prefiksowe, ale o stałej długości, więc łatwe do zdekodowania).

Kod binarny skrócony

- Tabela, którą za chwile zobaczymy przedstawia binarne kodowanie skrócone dla N=10 (wymaga czterech bitów, ale 6 kodów jest nieużywanych).
- W celu uzyskania kodowania skróconego przesuwamy nieużywane kody z końca tabeli do środka tak, aby pierwszy nieużywany kod występował po ilości kodów równej ilości kodów nieużywanych.

Kod binarny skrócony

Kodowanie binarne		Kodowanie binarne skrócone	
0	0000	0	0000
1	0001	1	0001
2	0010	2	0010
3	0011	3	0011
4	0100	4	0100
5	0101	5	0101
6	0110	-	0110
7	0111	-	0111
8	1000	-	1000
9	1001	-	1001
-	1010	-	1010
-	1011	-	1011
-	1100	6	1100
-	1101	7	1101
-	1110	8	1110
-	1111	9	1111

Kod binarny skrócony

- Jeżeli ilość kodów nieużywanych jest większa lub równa ¼ najbliższej potęgi dwójki po N, z kodów początkowych możemy usunąć wiodące zero, w przeciwnym wypadku zero można usunąć tylko z niektórych kodów tak, aby zachować prefiksowość kodu.
- W przykładzie następna potęga dwójki to 16, ilość nieużywanych kodów jest większa od 16·¼, a więc można usunąć wiodące zero z wszystkich kodów początkowych.
- Dla zwykłego kodowania binarnego średnia oczekiwana długość kodu to rzecz jasna 4, dla kodowania skróconego (przy założeniu równomiernej dystrybucji elementów alfabetu) wynosi ona (6·3 + 4·4) / 10 = 3,4.

Kodowanie Golomba

- Kod Golomba przewidziany jest (podobnie jak kodowanie unarne) dla geometrycznych rozkładów prawdopodobieństwa elementów alfabetu.
- Takie rozkłady spotyka się często w pozostałościach sygnału dźwiękowego po jego predykcji za pomocą filtru adaptacyjnego.
- Aby zmniejszyć wielkość alfabetu, wykorzystuje się arytmetykę modulo.

Kodowanie Golomba

- Oto przykład kodowania Golomba, niech M (baza kodu) będzie równe 10, a alfabet wejściowy składa się z liczb 8-bitowych ze znakiem, liczy więc sobie 256 elementów. Alfabet wyników z dzielenia będzie liczył 26 elementów, alfabet reszt z dzielenia 10 elementów.

Kodowanie Golomba

- 92 dodatnia -> 184 184 / 10 = 18 184 % 10 = 4

  1111111111111110 100

  kodowanie unarne kodowanie binarne skrócone

- -62 ujemna -> 125 125 / 10 = 12 125 % 10 = 5

  111111111110 101

- 13 dodatnia -> 26 26 / 10 = 2 26 % 10 = 6

  10 110

Kodowanie Rice'a

- Kod Rice'a to szczególny przypadek kodu Golomba, gdzie baza kodu M jest potęgą dwójki. Niesie to za sobą następujące korzyści:

1. Dzielenie może zostać zastąpione przez bitowe przesunięcie w prawo.
2. Operacja modulo może zostać zastąpiona przez maskę bitową (AND).
3. Ilość elementów w alfabecie reszt z dzielenia jest potęgą dwójki, koduje się je więc zwykłym kodem binarnym.

Kodowanie Rice'a

- Oto przykład kodowania kodem Rice'a, alfabet wiadomości to zakres liczb całkowitych od -128 do +127, baza kodu M = 16.

  słowo do zakodowania: 22 (dodatnie) -> 44,
  binarnie 00011100.

  Operacja dzielenia: 0001**1**100 >> 4 = 0001,
  kodowanie unarne daje „10”

  Operacja modulo: 0001**1100** & 00001111 = 1100,
  pozostaje w kodowaniu binarnym.

  Wynik: 10 1100

Kodowanie Rice'a

- Oto przykład kodowania kodem Rice'a, alfabet wiadomości to zakres liczb całkowitych od -128 do +127, baza kodu M = 16.

  słowo do zakodowania 7 (ujemne) -> 15,
  binarnie 00001111

  Operacja dzielenia: 0000**1111** >> 4 = 0000,
  kodowanie unarne daje „0”

  Operacja modulo: 0000**1111** & 00001111 = 1111, pozostaje w kodowaniu binarnym.

  Wynik: 0 1111

Zastosowania omówionych kodów

- Kody arytmetyczne

  Rzadko stosowane z dwóch przyczyn: po pierwsze spore wymagania pamięciowe i mocy obliczeniowej (to ograniczenie odchodzi już w przeszłość), duża ilość zastrzeżeń patentowych (to niestety nie). Często stosowane zamiast kodowania Huffmana jako ostatni stopień kompresji stratnej w nowszych kodekach (MPEG-4, AAC+).

Zastosowania omówionych kodów

- Kody Golomba i Rice'a

  Stosowane praktycznie tylko w wersji Rice'a, bezstratne kompresory audio: FLAC, ALAC (Apple), próby stosowania do bezstratnej kompresji obrazów (JPEG-LS).

Dziękuję za uwagę...

Damian Grela
e-mail: **dgrela@pk.edu.pl**
http://www.dgrela.pl