Wzory

- **Informacja:**
$$I(x) = -\log_2 P(x)$$
- **Entropia:**
$$H(X) = \sum P(x) \cdot I(x)$$
- **Śr. długość kodu:**
$$L = \sum P_i \cdot l_i$$
- **Nierówność Krafta:**
$$\sum 2^{-l_i} \leq 1$$
- **Wydajność:**
$$r = \frac{1}{n} \log |\mathcal{C}|$$

Przykłady:
$I(1/2) = 1, \quad H(\{1/2, 1/2\}) = 1, \quad L = 0.5 \cdot 1 + 0.5 \cdot 2 = 1.5, \quad \sum 2^{-l_i} = 0.75$

- **Hamming:** d = liczba różnic **Wykrywanie:** $d-1$, **Korekcja:** $\lfloor (d-1)/2 \rfloor$
- **Wsp. informacji:** $\eta = k/n$
- **Kod doskonały:**
$$\sum_{i=0}^{t} \binom{n}{i} \leq 2^{n-k}$$
- **Cykliczność:** przesunięcie $\Rightarrow$ w kodzie
- **Wielomian** $g(x)$**:**
$$g(x) \mid x^n + 1 \in \mathbb{Z}_2[x]$$

Kod Huffmana

Algorytm:

- Symbole a_i z wagami (P_i lub liczba wystąpień)
- Połącz 2 najmniejsze wagi $\rightarrow$ nowe drzewo
- Krawędzie: lewa = 0, prawa = 1
- Powtarzaj aż zostanie jedno drzewo
- Kody = ścieżki z korzenia

Wariant dynamiczny:

- Drzewo budowane w trakcie odczytu
- Nowe symbole $\Rightarrow$ użycie `NYT` (Not Yet Transmitted)
- Nowy symbol dodany do drzewa, reszta aktualizowana
- Znane symbole: kod z aktualnego drzewa

Kod prefiksowy, jednoznaczny, optymalny (min. L)

Kod Shannon-Fano

Kod prefiksowy, nie zawsze optymalny.

- Dla a_i z p_i: oblicz $l_i = \lceil -\log_2 p_i \rceil$
- Oblicz w_j: $\quad w_1 = 0, \quad w_j = \sum_{i=1}^{j-1} 2^{l_j - l_i}$
- Kod a_j: binarna postać w_j dopełniona zerami z lewej do l_j bitów

Przykład: $p = [0.5, 0.25, 0.25] \Rightarrow l = [1, 2, 2]$
$w = [0, 2, 3] \Rightarrow$ kody: 0, 10, 11

Kod Golomba/Rice

Dla liczby n i parametru M:
$$q = \lfloor \frac{n}{M} \rfloor, r = n \bmod M$$
Golomb: kod = q w unarnym + r w binarnym (długość zależna od M)
Rice: Golomb z $M = 2^k \Rightarrow r = k$-bitowy binarny

Kod Tunstalla

Kod nieprefiksowy, ale jednoznaczny przy stałej długości.
Kompresja bezstratna o stałej długości wyjściowej (kod: n bitów):

- Dla alfabetu a_i z p_i budujemy drzewo od najczęstszych symboli.
- Rozwijaj liść S: twórz ciągi $Sa_1, \ldots, Sa_m$ z wagą $P \cdot p_i$.
- Powtarzaj aż liczba liści (kodów) $= 2^n$.
- Kody to indeksy liści (słowa) $\Rightarrow$ długość kodu $= n$.

Przykład: $p(a) = 0.6$, $p(b) = 0.4 \Rightarrow$ start: a, b
Rozwijaj a: aa, ab itd. aż do np. 4 słów (dla $n = 2$)

Kodowanie Eliasa

Dla liczby całkowitej x, długości binarnej $n = \lfloor \log_2 x \rfloor + 1$:

- $\gamma(x) = 0^{n-1}\,(x)_2$
- $\delta(x) = \gamma(n) + (x)_2$ bez pierwszego bitu
- $\omega(x) = \omega(n-1) + (x)_2 + 0$ (rekurencja)

Przykład: $x = 9$, $(x)_2 = 1001$, $n = 4$
$\gamma(9) = 0001001 \quad \delta(9) = \gamma(4) + 001$
$\omega(9) = \omega(3) + 1001 + 0$

Kodowanie Fibonacciego

- Ciąg: $f_0 = f_1 = 1$, $f_n = f_{n-1} + f_{n-2}$ dla $n \geq 2$
- Reprezentacja liczby x: $x = \sum a_i f_i$, $a_i \in \{0, 1\}$
- Warunek: brak dwóch jedynek obok siebie (Zeckendorfa)
- Na końcu dodajemy dodatkowe 1 jako znacznik końca

Przykład: $x = 6$, $6 = 5 + 1 = f_4 + f_1 \Rightarrow$ `100100` + 1 = `1001001`

Kodowanie arytmetyczne

Kodowanie:

- $d = p - l$
- $l \leftarrow l + d \cdot F(j)$
- $p \leftarrow l + d \cdot F(j+1)$

Dekodowanie:

- $d = p - l$
- Znajdź j: $F(j) \leq \frac{x - l}{d} < F(j+1)$
- $l \leftarrow l + d \cdot F(j)$
- $p \leftarrow l + d \cdot F(j+1)$

Przykład: $F = [0, 0.5, 1]$, dla ciągu `AB`, A = 0–0.5, B = 0.5–1
Koniec przedziału: liczba z zakresu $(0.25, 0.75)$ reprezentuje słowo `AB`

Kodowanie słownikowe

LZ77

$(o, l, k) = \mathcal{C}_{i-o} \cdots \mathcal{C}_{i-o+l}\, k$

o — przesunięcie, l — długość, k — nowy znak
Przykład: `aabcaabc`
(0,0,a), (0,0,a), (0,0,b), (0,0,c), (4,3,c)

LZ78

Znajdź najdłuższy prefiks (lub ϵ)
Dodaj: prefiks + nowy znak
Zakoduj: (i, k)
Przykład: `ababcbababaaaaaaa`
(0,a), (0,b), (1,b), (0,c), (2,a), (4,a), (6,a), (7,a)

LZW

Startowy słownik = alfabet
Kodujemy indeksy ciągów (i)
Dodaj nowy ciąg gdy brak
Przykład: `abababa`
a (97), b (98), ab (256), ba (257), aba (258)
Kody: 97, 98, 256, 257, 258

BWT / bzip2

1. Dla słowa tworzymy wszystkie rotacje cykliczne. 2. Sortujemy je leksykograficznie. 3. Zapisujemy ostatnią kolumnę (BWT) i numer wiersza, gdzie znajduje się oryginalne słowo. 4. Odwracanie: znając BWT i indeks, możemy odzyskać oryginalne słowo.

e	h
h	o
ll	e
lo	l
o	l

0	1	2	3	4
e	h	l	l	o
2	0	3	4	1

Move-To-Front (MTF)

Transformacja zmniejszająca entropię (często po BWT).

- Start: tabela liter w porządku alfabetycznym
- Dla każdej litery: zapisz jej pozycję, przesuń ją na początek

Przykład: `hello` Alfabet: `[e, h, l, o]`

h $\rightarrow$ 1	tabela: [h, e, l, o]
e $\rightarrow$ 1	tabela: [e, h, l, o]
l $\rightarrow$ 2	tabela: [l, e, h, o]
l $\rightarrow$ 0	tabela: [l, e, h, o]
o $\rightarrow$ 3	tabela: [o, l, e, h]

Wynik: `11203`

PPM

- Dla symbolu sprawdzamy jego kontekst (max długość n).
- Przeglądamy konteksty: $n \rightarrow 0 \rightarrow -1$
- Dla $n \geq 0$: zliczamy wystąpienia; dla -1: tylko obecność.
- Budujemy model prawdopodobieństw — np. do Huffmana.

!: szukaj symbolu w najdłuższym kontekście. Jeśli nie ma – wypisz ESC i przejdź do krótszego.

Symbol
t
h
i
s
-

Symbol	Licznik
ESC	1
t	1
h	1
i	1
s	2
-	1

Kontekst	Symbol	Licznik
t	ESC	1
	h	1
h	ESC	1
	i	1
i	ESC	1
	s	2
s	ESC	1
	-	1
-	ESC	1
	i	1

Kontekst	Symbol	Licznik
th	ESC	1
	i	1
hi	ESC	1
	s	1
is	ESC	1
	-	1
s-	ESC	1
	i	1
-i	ESC	1
	s	1

Kody Hamminga

- Długość kodu: $n = 2^m - 1$, dane: $k = n - m$
- Macierz parzystości H: kolumny = binarne zapisy $1, \ldots, n$
$$H_H(3) = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$
- Macierz generująca G: $[I_k \mid P]$, gdzie P pochodzi z H
$$G_H(3) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Rozszerzony kod Hamminga: dodany bit parzystości – długość $n+1$ **Cykliczny kod Hamminga:** generowany przez $g(x)$, który dzieli $x^n + 1$ w $\mathbb{Z}_2[x]$ **Kod Hamminga (7,4)** jest kodem doskonałym dla $d = 3$

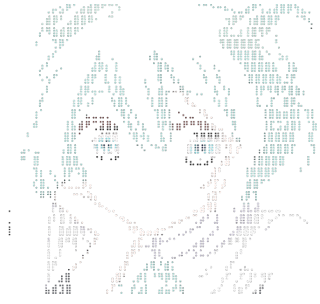

Fig. 1: Exam Emotional Support System
Hatsune przypomina: Nie da się skompresować wszystkich możliwych słów — zbiór skompresowanych musi być mniejszy => nie istnieje bijekcja.

Lew: Twoja pewność siebie jest jak kod Huffmana — konkretna i skuteczna.

Ryby: Chaos w danych? Nie szkodzi — PPM poradzi sobie nawet z tobą.

Wodnik: BWT to twoja dusza — nieoczywista, ale piękna w dekodowaniu.

Koziorożec: Twoja determinacja przypomina kod Hamminga — wykrywa, poprawia i nie odpuszcza.

☐☐☐☐☐

Fig. 2: Zakreśl, jeśli: *Błąd w obliczeniach? Brak pomysłu? Nie rozumiesz pytania?*

"Wszystko da się skompresować. Poza czasem na egzaminie."
– autor nieznany, ale na pewno sfrustrowany