Kompresja danych DKDA (2)

Marcin Gogolewski
marcing@wmid.amu.edu.pl

Uniwersytet im. Adama Mickiewicza w Poznaniu

Poznań, 10 października 2025

Na poprzednim wykładzie

- rodzaje i metody kompresji
- miary kompresji
- autoinformacja (jednostki)
- entropia (dla modelu probabilistycznego)
- kodowanie jednoznaczne, kody prefiksowe
- kodowanie Huffmana (statyczne)

Definicja problemu

Jak kodować liczby naturalne?

Szukamy sposobu kodowania dowolnej liczby naturalnej (całkowitej, większej od zera) za pomocą kodu o zmiennej długości. Liczba wykorzystanych bitów powinna być proporcjonalna do kodowanej wartości (asymptotycznie $\Theta(\log_2 n)$).

Zastosowanie

Czasami konieczne jest przesłanie kilku liczb, których wielkość ciężko oszacować (wykorzystywane np. przez niektóre odmiany LZ77 do kodowania przesunięć).

Kodowanie Eliasa

Kodowanie Eliasa (lata '70)

Wyróżniamy trzy rodzaje kodów Eliasa (co najmniej trzy), różniące się konstrukcją: γ, δ, ω. Ogólna idea jest następująca:

- kodujemy zadaną liczbę binarnie;
- „doklejamy" informację o długości tak, by powstał kod jednoznacznie dekodowalny.

Kodowanie γ

- kodowana liczba x jest zapisywana w kodzie binarnym,
- niech n oznacza liczbę cyfr znaczących w zapisie binarnym,
- słowo składa się z:
  1. $(n-1)$ bitów o wartości 0,
  2. liczby x zapisanej binarnie.

Przykład

Niech $x = 137_{10}$. Zatem $x = 10001001_2$, $n = 8$, a wynikowy kod to:

$$\underbrace{0000000}_{7\times 0}\underbrace{10001001}_{x \text{ binarnie}}.$$

Słowo kodowe ma w tym przypadku 15 bitów.

Kodowanie δ

- liczba x jest zapisywana w kodzie binarnym,
- niech n oznacza liczbę bitów binarnej reprezentacji x,
- liczbę n przedstawiamy w kodzie γ,
- z binarnej reprezentacji liczby x usuwamy najbardziej znaczącą cyfrę,
- niech k oznacza liczbę bitów znaczących n,
- wynikowym kodem jest:
  - $k-1$ zer,
  - binarna reprezentacja liczby n,
  - binarna reprezentacja liczby x z usuniętą pierwszą jedynką.

Liczba bitów reprezentacji to:

$$\underbrace{2\lceil \log_2(\lceil \log_2 x \rceil)\rceil - 1}_{n \text{ i } k-1} + \underbrace{\lceil \log_2 x \rceil - 1}_{x}.$$

Przykład kodowania δ

Przykład

Niech $x = 137_{10}$. Zatem $x = 10001001_2$, $n = 8 = 1000_2$, a $k = 4$. Wynikowy kod (niech $x_{(-1)}$ to x po usunięciu najbardziej znaczącej jedynki) to:

$$\underbrace{000}_{3\times 0}\,\underbrace{1000}_{n}\,\underbrace{0001001}_{x_{(-1)}}.$$

Kodowanie ω

Kodowanie

Algorytm wygląda następująco (kolejne podsłowa binarne dołączamy na początek powstającego kodu):

1. zapisz bit 0 (znacznik końca),
2. $k \leftarrow x$,
3. while $k > 1$:
   1. zapisz dwójkowo liczbę k,
   2. nowe k, to liczba bitów k z poprzedniego kroku, pomniejszona o 1.

Liczba bitów kodu wynosi (w każdym kroku zapisujemy $k_i = \lceil \log_2 k_{i-1} \rceil$ bitów):

$$1 + \sum_{i=1}^{n} k_i,$$

gdzie n jest liczbą kroków.

Dekodowanie ω

Dekodowanie

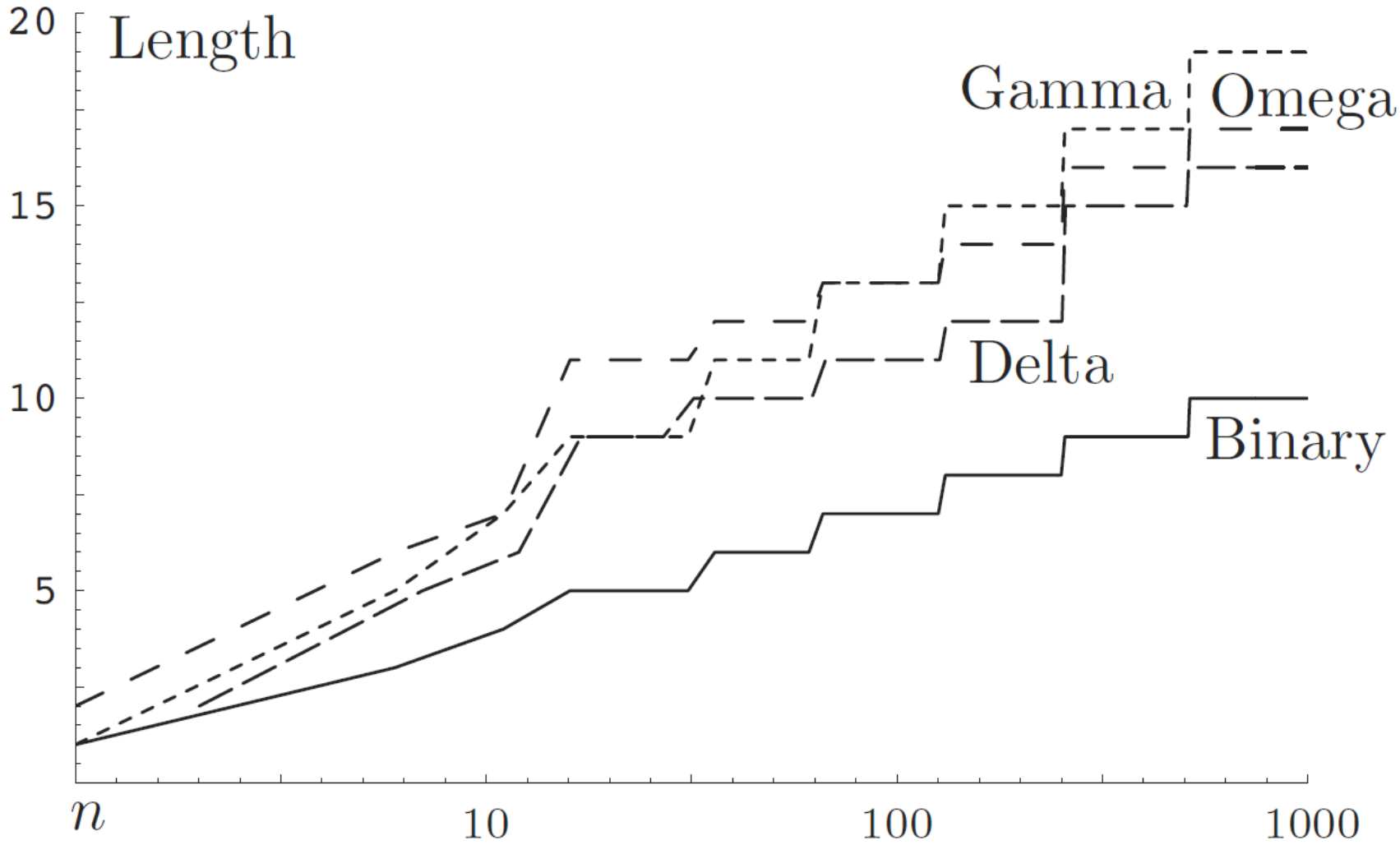
1. $n \leftarrow 1$,
2. while „kolejny bit nie ma wartości 0”:
   1. $n \leftarrow$ wartość zapisana na kolejnych $n+1$ bitach,
3. $x \leftarrow n$.

Przykład kodowania

Niech $x = 109_{10}$.

krok	k	słowo binarne	liczba bitów	kod
1		0	1	0
2	109	1101101	7	1101101 0
3	6	110	3	110 1101101 0
4	2	10	2	10 110 1101101 0
5	1			

Porównanie kodowań: γ, δ i ω



Kody Fibonacciego

Liczby Fibonacciego

Liczby postaci: $f_0 = f_1 = 1$, $f_n = f_{n-1} + f_{n-2}$ dla $n \geq 2$.

Jedna z własności liczb Fibonacciego

Każda liczba całkowita dodatnia k może być przedstawiona jednoznacznie jako: $k = \sum_{i=1}^{\infty} a_i f_i$, gdzie a_i jest równe 0 lub 1.
Obserwacja: zawsze możemy zapisać ten ciąg w taki sposób, że nie wystąpią obok siebie dwie jedynki!

Algorytm kodujący

- znajdujemy współczynniki a_i,
- zapisujemy w kolejności od najmniej znaczącego,
- na końcu dopisujemy jedynkę.

(na końcu zawsze wystąpią dwie jedynki!)

Kodowanie Fibonacciego (2)

Przykładowe liczby

liczba	kod binarny		liczba	kod binarny
1	11		7	01011
2	011		8	000011
3	0011		9	100011
4	1011		10	010011
5	00011		11	001011
6	10011		12	101011

Algorytm dekodowania i oszacowanie długości kodu – do przemyślenia.

Entropia źródła

Dla źródła danych $\mathcal{S}$ generującego ciąg $\{X_1, \ldots, X_n\}$ nad alfabetem $\mathcal{A} = \{1, 2, \ldots, m\}$ entropię definiujemy jako:

$$H(S) = \lim_{n \to \infty} \frac{1}{n} G_n \ , \tag{1}$$

gdzie

$$G_n = -\sum_{i_1=1}^{m} \ldots \sum_{i_n=1}^{m} \Pr[X_1 = i_1, \ldots, X_n = i_n] \log \Pr[X_1 = i_1, \ldots, X_n = i_n] \ .$$

Entropia pierwszego rzędu

Jeżeli wszystkie elementy ciągu mają identyczny i niezależny rozkład, to można pokazać, że:

$$G_n = -n \sum_{i_1=1}^{m} \Pr[X_1 = i_1] \log \Pr[X_1 = i_1] \ .$$

Wtedy wzór na entropię przyjmuje postać:

$$H(S) = -\sum_{i_1=1}^{m} \Pr[X_1 = i_1] \log \Pr[X_1 = i_1] \ . \tag{2}$$

Jeżeli chcemy rozróżnić powyższe wartości (tzn. warunek o rozkładzie i niezależności nie jest spełniony), to entropię 2 nazywamy entropią pierwszego rzędu źródła, a 1 po prostu entropią źródła.

Model fizyczny

Do stworzenia modelu wykorzystujemy wiedzę o fizycznych właściwościach źródła.
Problem: charakterystyka źródła jest zwykle nieznana (lub zbyt skomplikowana do bezpośredniego wykorzystania).

Modele probabilistyczne

Najprostszy model

Zakładamy, że każda litera alfabetu występuje z jednakowym prawdopodobieństwem.

Niezależność, ale różne prawdopodobieństwa

Każdej literze przyporządkowujemy pewne prawdopodobieństwo, jednak (zakładamy) kolejne litery są niezależne od siebie (założenie wykorzystywane przy kodowaniu Huffmana, arytmetycznym, ...). Kody mogą być bardzo efektywne, jeżeli założenie jest rzeczywiście spełnione.

Dyskretny łańcuch Markowa – przypomnienie

Niech $\{x_n\}$ będzie ciągiem obserwacji, jeżeli:

$$\Pr[x_n|x_{n-1}, x_{n-2}, \ldots, x_{n-k}] = \Pr[x_n|x_{n-1}, x_{n-2}, \ldots, x_{n-k}, \ldots] \ ,$$

to ciąg ten spełnia model Markowa k-tego rzędu (wiedza o k ostatnich elementach równoważna jest wiedzy o całości procesu). Wartości x_i nazywamy stanami procesu.
Najczęściej wykorzystujemy model pierwszego rzędu.

Model Markowa

Przykład 1

Jeżeli założymy, że zależność jest liniowa, to możemy przyjąć model w którym symbol generowany przez źródło jest zależny od poprzedniego:

$$x_n = \rho x_{n-1} + \varepsilon_n \ .$$

Kodujemy wyłącznie ciąg $\{\varepsilon_1, \ldots, \varepsilon_n\}$ (i, być może, także wartość początkową x_0).

Model Markowa

Przykład 2

Kodujemy obraz binarny. Nasz proces ma dwa stany S_c (piksel czarny) oraz S_b piksel biały (zależność nie jest liniowa!).

$$\Pr(c|c) \neq \Pr(c|b) \ .$$

W tym przypadku entropia jest średnią (ważoną) entropii w poszczególnych stanach:

$$H = \sum_{i=1}^{M} \Pr(S_i) H(S_i) \ .$$

Zastosowania

- kompresja tekstu (algorytmy słownikowe)
- kompresja obrazu
- kompresja dźwięku (np. mowy)

Model mieszany

Problem w kompresji tekstu ciężko bezpośrednio stosować model Markowa ze względu na ogromną liczbę stanów.

Rozwiązanie model Markowa jest wykorzystywany tylko w przypadkach o stosunkowo wysokim prawdopodobieństwie, w pozostałych jest wykorzystywany np. model probabilistyczny (przykład: algorytm PPM).

Nierówność Krafta-McMillana – przypomnienie

Niech C będzie kodem składającym się z N słów o długościach $l_1, l_2, \ldots, l_N$. Jeżeli C jest jednoznacznie dekodowalny, to

$$K(C) = \sum_{i=1}^{N} \frac{1}{2^{l_i}} \leqslant 1.$$

Czy można skonstruować optymalny kod przy podanych długościach słów?

Kody Shannon-Fano

- Niech symbole a_i występują odpowiednio z prawdopodobieństwami p_i.
- Weźmy długości kodów $l_i = \lceil -\log p_i \rceil$.
- Długości l_i spełniają nierówność Krafta-McMillana.

$$\sum_{i=1}^{N} \frac{1}{2^{l_i}} \leqslant \sum_{i=1}^{N} \frac{1}{2^{-\log p_i}} = \sum_{i=1}^{N} p_i = 1 \ .$$

- Istnieje więc kod prefiksowy o takich długościach.
- Łatwo zauważyć, że średnia długość tego kodu jest nie większa niż entropia plus 1.

Konstrukcja kodu o podanych długościach

- Niech $l_1 \leqslant l_2 \leqslant \ldots \leqslant l_N$.
- Definiujemy pomocnicze $w_1, w_2, \ldots, w_N$ jako

$$w_1 = 0 \qquad w_j = \sum_{i=1}^{j-1} 2^{l_j - l_i}$$

- Binarna reprezentacja w_j dla $j > 1$ zajmuje $\lceil \log w_j \rceil$ bitów.
- Liczba bitów w_j jest mniejsza lub równa l_j. Dla w_1 to oczywiste.

$$\begin{aligned} \log w_j &= \log \left[\sum_{i=1}^{j-1} 2^{l_j - l_i} \right] = \log \left[2^{l_j} \sum_{i=1}^{j-1} 2^{-l_i} \right] = \\ &= l_j + \log \left[\sum_{i=1}^{j-1} 2^{-l_i} \right] \leqslant l_j \end{aligned}$$

Konstrukcja kodu o podanych długościach

Kodowanie wygląda następująco:

Jeżeli $\lceil \log w_j \rceil = l_j$, to j-te słowo kodowe jest binarną reprezentacją w_j. Jeżeli jest mniejsze ($\lceil \log w_j \rceil < l_j$), to reprezentację w_j uzupełniamy odpowiednią liczbą zer z lewej strony (kod liczby $l_j - \lceil \log w_j \rceil$).

Konstrukcja kodu o podanych długościach

Czy to jest kod prefiksowy?

Załóżmy, że c_j jest prefiksem c_k i $j < k$. Wtedy l_j pierwszych bitów c_k tworzy c_j, czyli $w_j = \left\lfloor \frac{w_k}{2^{l_k - l_j}} \right\rfloor$,

ale $w_k = \sum_{i=1}^{k-1} 2^{l_k - l_i}$.

Czyli

$$\begin{aligned} \frac{w_k}{2^{l_k - l_j}} &= \sum_{i=1}^{k-1} 2^{l_j - l_i} = w_j + \sum_{i=j}^{k-1} 2^{l_j - l_i} = \\ &= w_j + 2^0 + \sum_{i=j+1}^{k-1} 2^{l_j - l_i} \geqslant w_j + 1 \ . \end{aligned}$$

Sprzeczne z założeniem, że c_j jest prefiksem c_k.

Przykład

- Weźmy a, b, c, d z prawdopodobieństwami $\frac{1}{3}, \frac{1}{4}, \frac{1}{4}, \frac{1}{6}$.
- Odpowiednio długości kodów Shannon-Fano wynoszą $2, 2, 2, 3$.
- Wyliczamy $w_a = 0$, $w_b = 1$, $w_c = 2$, $w_d = 6$.
- Kody to odpowiednio
  $kod(a) = 00$, $kod(b) = 01$,
  $kod(c) = 10$, $kod(d) = 110$.

Problemy z kodami o zmiennej długości

- propagacja błędów (przekłamanie jednego bitu może zniszczyć całą transmisję)
- problemy z przetwarzaniem (głównie sprzętowym)
- pomysł: kody o stałej długości kodujące zmienną liczbę symboli

Warunki na kod o stałej długości

- każdy ciąg wejściowy powinien dać się przekształcić na ciąg symboli z książki kodowej
- należy zmaksymalizować średnią liczbę symboli (z pierwotnego alfabetu) na słowo kodowe

Algorytm Tunstalla

- Chcemy stworzyć n-bitowy kod dla źródła generującego symbole z N-literowego alfabetu.
- Na początku książka kodowa zawiera wszystkie litery alfabetu źródłowego.
- Usuwamy element o największym prawdopodobieństwie i dodajemy wszystkie możliwe pary postaci `(element, litera alfabetu)`. Zakładamy niezależność zdarzeń, zatem prawdopodobieństwa łączne są iloczynami.
- Powtarzamy czynność K razy, za każdym razem zwiększając rozmiar słownika o $N-1$ ($N + K(N-1) \leq 2^n$).

Przykład

- pierwotny alfabet i prawdopodobieństwa wystąpień
  A(0,6), B(0,3), C(0,1)
- po pierwszej iteracji
  B(0,3), C(0,1), AA(0,36), AB(0,18), AC(0,06)
- po drugiej iteracji – kody
  B – 000, C – 001, AB – 010, AC – 011, AAA – 100, AAB – 101, AAC – 110

Problem i przykład rozwiązania

Problem jak zakodować dowolną liczbę całkowitą, przy założeniu, że im większa liczba, tym mniejsze prawdopodobieństwo jej wystąpienia?

Przykład rozwiązania dla $\mathsf{Pr}[k] = \frac{1}{2^k}$ (nieskończony alfabet): kod unarny (k jedynek zakończone zerem – np. 4 – 11110).

Obserwacja dla powyższego modelu kod ten jest równoważny z kodem Huffman'a (a zatem optymalny!).

Kod Golomba (rodzina kodów)

- rodzina kodów sparametryzowana liczbą całkowitą $m > 0$
- każda liczba n zapisywana jest za pomocą pary liczb q i r
- $q = \lfloor \frac{n}{m} \rfloor$ (zapisywana za pomocą kodu unarnego)
- $r = n - q \cdot m$ (zapisywana za pomocą $\lfloor \log_2 m \rfloor$ lub $\lceil \log_2 m \rceil$ bitów)

Kod Golomba – przykład dla $m = 5$

n	q	r	Słowo kodowe	n	q	r	Słowo kodowe
0	0	0	000	8	1	3	10110
1	0	1	001	9	1	4	10111
2	0	2	010	10	2	0	11000
3	0	3	0110	11	2	1	11001
4	0	4	0111	12	2	2	11010
5	1	0	1000	13	2	3	110110
6	1	1	1001	14	2	4	110111
7	1	2	1010	15	3	0	111000

Dynamiczne kody Huffmana

- Aby wykonać kodowanie Huffmana musimy znać prawdopodobieństwa (częstość występowania) liter.
- Co zrobić, gdy dane napływają na bieżąco i nie znamy statystyk?
- Kodować $k+1$ symbol na podstawie statystyk k symboli.

Przygotowania ($\star$)

- Dla alfabetu wejściowego mamy kodowanie stałej długości (pomocnicze).
- Dla kodu Huffmana tworzymy na bieżąco jego drzewo o następujących własnościach:
  - Każdy liść odpowiada symbolowi i zawiera wagę – liczbę dotychczasowych wystąpień.
  - Wierzchołki wewnętrzne mają wagę będącą sumą wag liści z poddrzew.
  - Każdy wierzchołek drzewa ma unikalny numer x_i. Numery te tworzą porządek zgodny z wagami wierzchołków (większa waga to większy numer wierzchołka). Dodatkowo rodzeństwo ma zawsze dwa kolejne numery.
- Na początku drzewo zawiera jeden wierzchołek o wadze 0 i etykiecie NYT oznaczającej, że symbol nie był jeszcze przesyłany.

Opis algorytmu

Pierwsze wystąpienie symbolu a

- Wyślij kod NYT i kod stałej długości dla nowego symbolu.
- Stary NYT podziel na dwa wierzchołki potomne – nowy NYT i liść a, nadaj a wagę 1. Nadaj im odpowiednie numery.
- Zmodyfikuj drzewo dodając 1 do wierzchołków wewnętrznych na ścieżce od a do korzenia i przebudowując drzewo tak, aby było zgodne z warunkami z $(\star)$.

Opis algorytmu

Kolejne wystąpienie symbolu a

- Znajdź liść a i wyślij odpowiadający mu kod.
- Zwiększ wagę a o 1.
- Zmodyfikuj drzewo dodając 1 do wierzchołków wewnętrznych na ścieżce od a do korzenia i przebudowując drzewo tak, aby było zgodne z warunkami z $(\star)$.

Modyfikacja drzewa

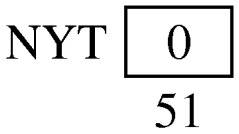
- Zbiór wierzchołków o tej samej wadze nazywamy blokiem.
- Jeśli pierwszy wierzchołek od dołu nie ma największego numeru w swoim bloku to zamieniamy go z tym o największym numerze odpowiednio przebudowując drzewo z zachowaniem własności. Następnie aktualizujemy wagę i patrzymy dalej rekurencyjnie.
- Kończymy jak dojdziemy do korzenia.

Przykład

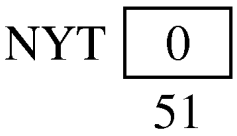
- Mamy cztery litery z 26 (zatem w drzewie maksymalnie 51 wierzchołków).

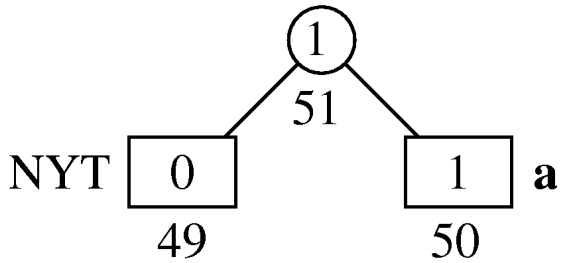
a	d	r	v
00001	00100	10010	10110

- Drzewo początkowo wygląda tak



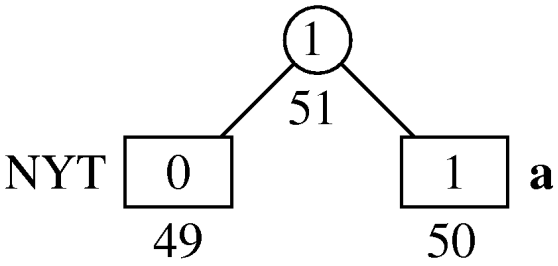
Przykład

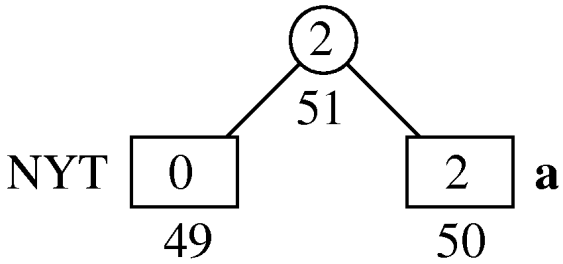




- Pojawia się litera *a*.
- Wysyłamy kod NYT (ε) i stały kod *a*: 00001.
- Modyfikujemy drzewo.

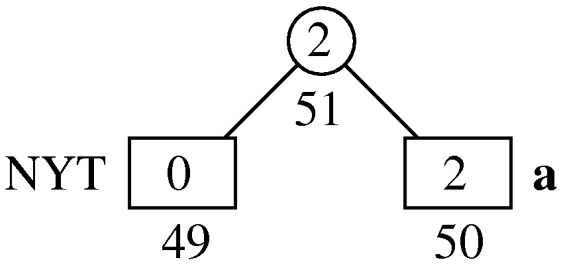
Przykład

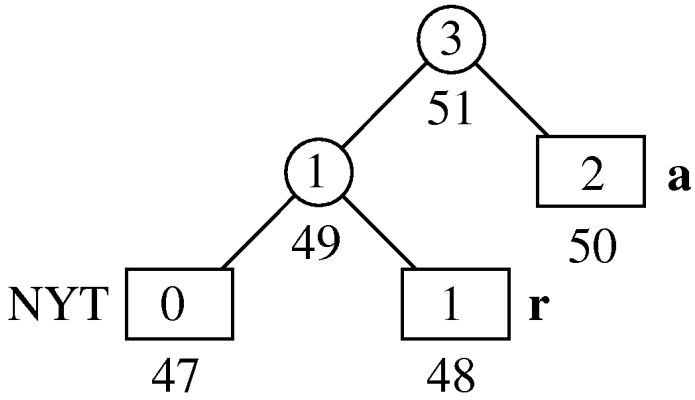




- Pojawia się druga litera *a*.
- Wysyłamy kod *a*: 1 (z drzewa lewe krawędzie to 0, a prawe to 1).
- Modyfikujemy drzewo.

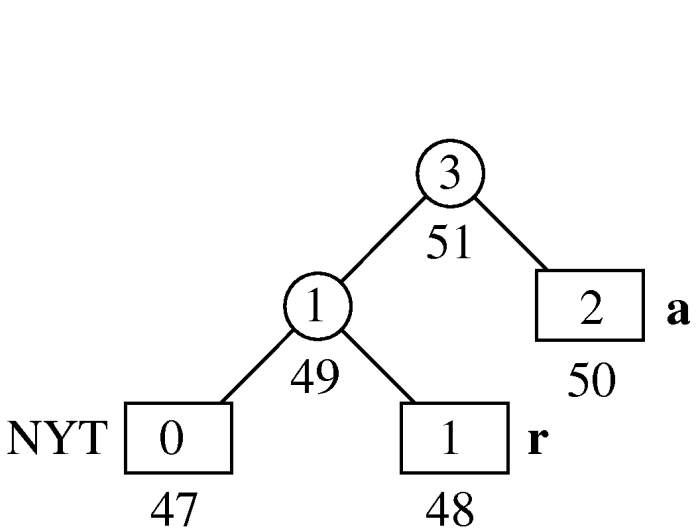
Przykład

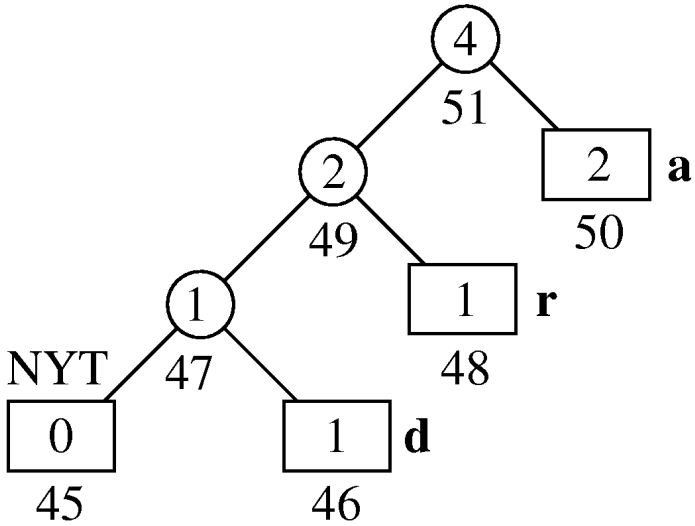




- Pojawia się litera *r*.
- Wysyłamy kod NYT (0) i stały kod *r*: 010010.
- Modyfikujemy drzewo.

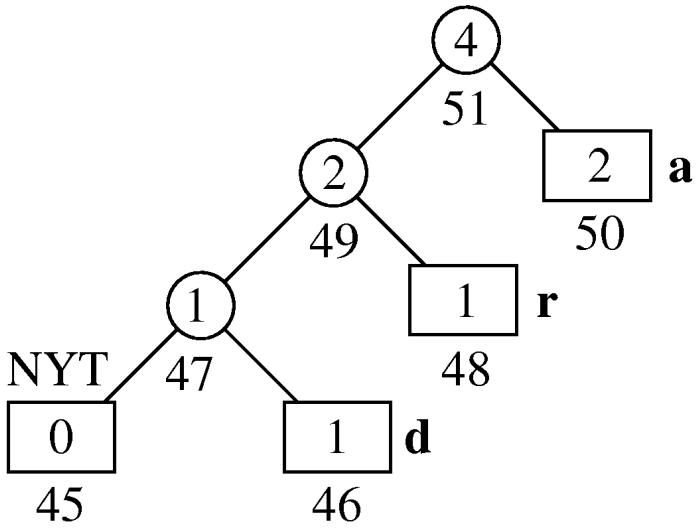
Przykład





- Pojawia się litera *d*.
- Wysyłamy kod NYT (00) i stały kod *d*: 0000100.
- Modyfikujemy drzewo.

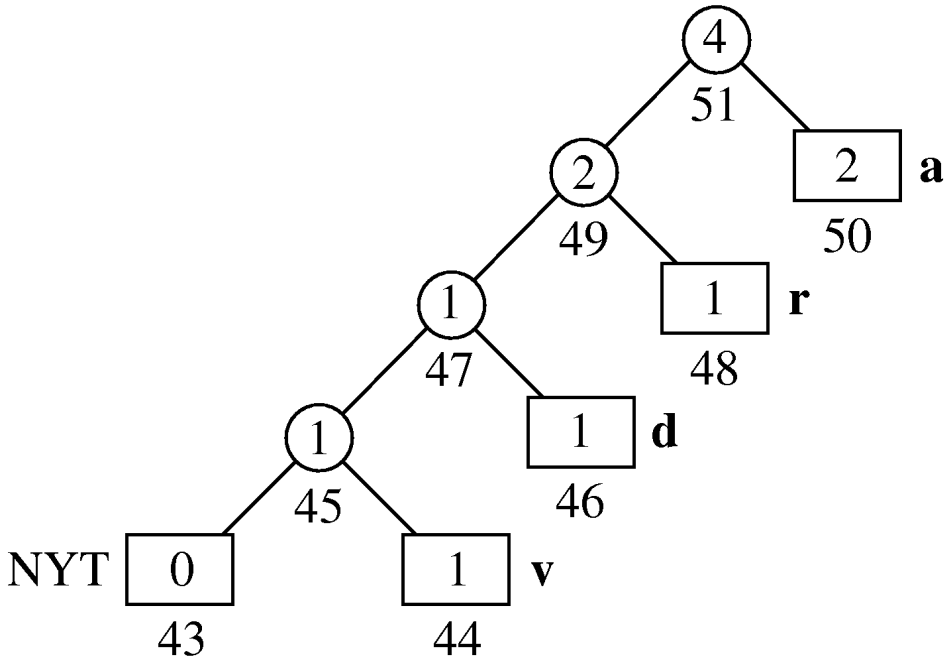
Przykład

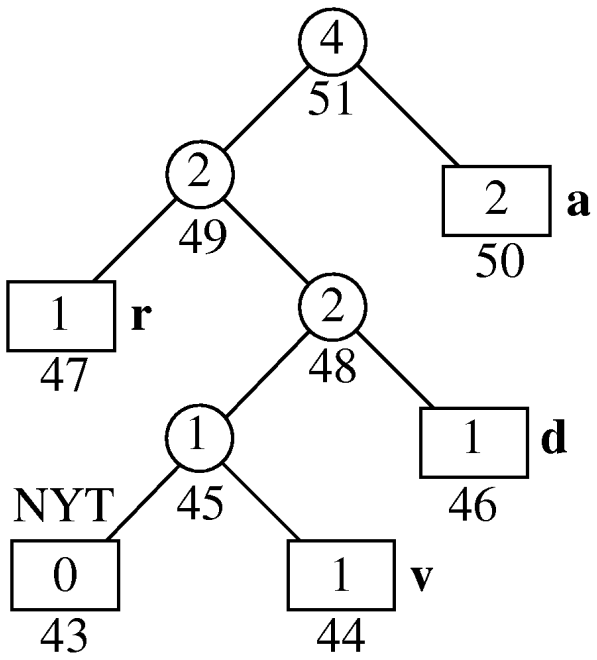


- Pojawia się litera v.
- Wysyłamy kod NYT (000) i stały kod v: 00010110.

Przykład

- Modyfikujemy drzewo (zamieniamy poddrzewa):





Przykład

- Modyfikujemy drzewo (zamieniamy poddrzewa):