Teoria Informacji i Kodowanie

Michał Antkowiak

Zakład Fizyki Komputerowej
Wydział Fizyki
Uniwersytet im. Adama Mickiewicza

Plan wykładu

Wstęp

Notacja i kody

Pojęcie entropii

Optymalne kodowanie - kod Huffmana

Optymalne kodowanie - kod Shannona-Fano

Kodowanie słownikowe

Kodowanie arytmetyczne

Wstęp

- Wprowadzenie w teorię przydatną w wielu zastosowaniach informatyki (m.in. w kryptografii, w modelowaniu języka naturalnego, czy w bio-informatyce).
- Teoria ta okresla ilościowo informację zawartą w zmiennej losowej lub w ciągu bitów, a także kryteria optymalnego przesyłania zakodowanej wiadomości przez zaszumiony kanał.

- Wykład (15 h) + ćwiczenia (15 h)

Literatura

- Information and Coding Theory, Gareth A. Jones and J. Mary Jones, Springer, 2000.
- Elements of Information Theory, Thomas M. Cover and Joy A. Thomas, Wiley Series in Telecommunications, 1991.
- An Introduction to Kolmogorov Complexity and Its Applications, Ming Li and Paul Vitanyi, Springer, 1997.
- Information Theory, Inference, and Learning Algorithms, David J.C. MacKay, Cambridge University Press, 2003.
- Information Theory and Network Coding, Raymond Yeung, Springer, 2008.

- http://wazniak.mimuw.edu.pl/index.php?title=Teoria_informacji

Notacja

- Słowa vs. liczby
- Celowa redundancja
  - kwota słownie
  - literowanie
  - alfabet fonetyczny ICAO

- Teoria informacji charakteryzuje w sposób matematyczny zapis, przesyłanie i odtwarzanie informacji
- Dąży do pogodzenia dwóch przeciwstawnych celów:
  - zapisywania wiadomości jak najzwięźlej
  - chronienia wiadomości przed przekłamaniami podczas transmisji

Notacja

- Czy istnieją wiadomości, których nie da się zapisać zwięźlej?
- Paradoks Berry'ego
  - Bertrand Russell
  - *Niech n będzie najmniejszą liczbą naturalną której nie da się zdefiniować w języku polskim za pomocą mniej niż dwudziestu pięciu słów.*
  - paradoks nieciekawej liczby
- Definicja nie może być częścią opisywanego obiektu
- Notacją dla S nazywamy dowolną różnowartościową funkcję $\alpha : S \to \Sigma^*$, gdzie Σ jest skończonym alfabetem.

Notacja

- Jeśli $|S| = m \geqslant 1$ i $|\Sigma| = r \geqslant 2$, to dla pewnego $s \in S$ zachodzi: $|\alpha(s)| \geqslant \lfloor \log_r m \rfloor$
- Jeśli $\alpha : N \to \Sigma^*$ jest dowolną notacją dla liczb naturalnych, to dla nieskończenie wielu n musi zajść: $\alpha(n) > \lfloor \log_r n \rfloor$
- Istnieje nieskończenie wiele liczb pierwszych (Euklides)

Kody

- Praktyczne zastosowanie niektórych własności notacji
- Odwzorowanie $\varphi : S \to \Sigma^*$ dla skończonego niepustego zbioru S jest kodem, jeśli jego naturalne rozszerzenie do morfizmu $\hat{\varphi}$ jest różnowartościowe. Mówimy, że kod jest bezprefiksowy, jeśli dodatkowo $\hat{\varphi}(s) \not\leq \hat{\varphi}(s')$ dla $s \neq s'$
- Każdy zbiór bezprefiksowy jest kodem.
- Kod, który nie jest bezprefiksowy: $\varphi(S) = \{aa, baa, ba\}$
- Notacja, która nie jest kodem: $\varphi(S) = \{a, ab, ba\}$
- Gra w 20 pytań

Kody bezprefiksowe jako drzewa

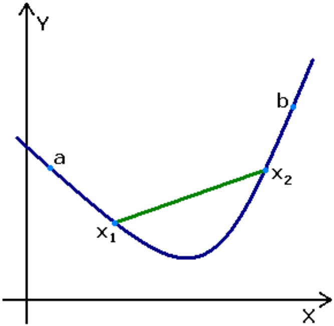
- Drzewo nad zbiorem X (X-drzewo) - dowolny niepusty zbiór $T \subseteq X^*$ zamknięty na operację brania prefiksu
- Dla dowolnego $w \in T$, $x \in X$ i $v \in X^*$:
  - wx jest bezpośrednim następnikiem (lub dzieckiem) w
  - wv jest poniżej w
  - zbiór $T_w = \{v : wv \in T\}$ nazywamy poddrzewem indukowanym przez w.
- Dowolny bezprefiksowy kod $\varphi : S \to \Sigma^*$ indukuje drzewo nad $\Sigma : T_\varphi = \{w : \exists s : w \leqslant \varphi(s)\}$
- Dowolne drzewo $T \subset \Sigma^*$ z $|S|$ liśćmi indukuje bezprefiksowy kod nad S

Nierówność Krafta

- Dla danego kodu $\varphi : S \to \Sigma^*$, niech $|\varphi| : S \to \mathbb{N}$ określa funkcję długości, zdefiniowaną jako $|\varphi|(s) = |\varphi(s)|$
- Niech $2 \leqslant |S| < \infty$ i niech $|\Sigma| = r$. Funkcja $\ell : S \to \mathbb{N}$ jest funkcją długości ($\ell = |\varphi|$) dla pewnego bezprefiksowego kodu $\varphi : S \to \Sigma^*$ wtedy i tylko wtedy, gdy $\sum_{s \in S} \frac{1}{r^{\ell(s)}} \leqslant 1$
- Nierówność Krafta-McMillana jest warunkiem koniecznym, który musi spełniać kod, aby był jednoznacznie dekodowalny
- Jeśli kod nie jest bezprefiksowy, nierówność Krafta wciąż jest spełniona
- Dla dowolnego kodu $\varphi : S \to \Sigma^*$ istnieje bezprefiksowy kod φ' dla którego $|\varphi| = |\varphi'|$ (McMillan)

Własności funkcji wypukłych

- Funkcja $f : [a, b] \to \mathbb{R}$ jest wypukła (na [a,b]) jeśli $\forall x_1, x_2 \in [a, b]$, $\forall \lambda \in [0, 1]$: $\lambda f(x_1) + (1 - \lambda) f(x_2) \geqslant f(\lambda x_1 + (1 - \lambda) x_2)$



- Jeśli f jest ciągła na [a,b] i dwukrotnie różniczkowalna na (a,b) oraz $f'' \geqslant 0 (f'' > 0)$, to jest wypukła (ściśle wypukła).

Własności funkcji wypukłych

- Określając X jako zmienną losową na S, zawsze będziemy zakładać, że S jest dana razem z rozkładem prawdopodobieństwa $p : S \to [0,1]$ (a więc $\sum_{s \in S} p(s) = 1), iX : S \to \mathbb{R}$.
- Wartość oczekiwana zmiennej X to $EX = \sum_{s \in S} p(s) \cdot X(s)$
- Jeśli $S = \{s_1, \ldots, s_m\}$, będziemy używać notacji $p_i = p(s_i)$, $x_i = X(s_i)$. W takim zapisie $EX = p_1 x_1 + \ldots + p_m x_m$. Od razu zauważmy, że EX nie zależy od tych x_i, dla których $p_i = 0$. Mówimy, że X jest stała, jeśli $p_i > 0$ zachodzi tylko dla jednej wartości i.
- Nierówność Jensena: Jeśli $f : [a, b] \to \mathbb{R}$ jest funkcją wypukłą, to dla każdej zmiennej losowej $X : S \to [a, b]$, $Ef(X) \geqslant f(EX)$. Jeśli dodatkowo f jest ściśle wypukła, to powyższa nierówność jest ścisła z wyjątkiem sytuacji, gdy X jest stała.

Własności funkcji wypukłych

- Aby nie rozważać za każdym razem szczególnych przypadków, przyjmiemy konwencję $0 \log_r 0 = 0 \log_r \frac{1}{0} = 0$
- Jest to uzasadnione przejściami granicznymi: $lim_{x \to 0^+} x \log_r x = \lim_{x \to 0^+} -x \log_r \frac{1}{x} = \lim_{y \to \infty} -\frac{\log_r y}{y} = 0.$
- W dalszej części wykładu przydatna będzie funkcja $x \log_r x$. Na podstawie lematu powyżej łatwo pokazać, że dla r>1 funkcja ta jest ściśle wypukła na przedziale $[0, \infty)$, mamy bowiem: $(x \log_r x)'' = \left(\log_r x + x \cdot \frac{1}{x} \cdot \log_r e\right)' = \frac{1}{x} \cdot \log_r e > 0$
- Złoty lemat: Niech $1 = \sum_{i=1}^q x_i \geqslant \sum_{i=1}^q y_i$, gdzie $x_i \geqslant 0$ i $y_i > 0$ dla $i = 1, \ldots, q$ i niech $r > 1$. Wtedy $\sum_{i=1}^q x_i \cdot \log_r \frac{1}{y_i} \geqslant \sum_{i=1}^q x_i \cdot \log_r \frac{1}{x_i}$ i równość zachodzi tylko wtedy, gdy $x_i = y_i$ dla $i = 1, \ldots, q$.

Entropia

- Entropią przestrzeni probabilistycznej S (parametryzowaną przez $r > 1$) nazywamy funkcję

$$\begin{aligned} H_r(S) &= \sum_{s \in S} p(s) \cdot \log_r \frac{1}{p(s)} \\ &= -\sum_{s \in S} p(s) \cdot \log_r p(s) \end{aligned}$$

- $H_r(S)$ jest wartością oczekiwaną zmiennej losowej zdefiniowanej na S jako $s \mapsto \log_r \frac{1}{p(s)}$
- Zwykle będziemy przyjmować $r = 2$, dlatego H będzie równoznaczne z H_2

Entropia

- Definicja entropii łączy dwa pomysły:
  - wyliczenie wartości oczekiwanej pewnej funkcji złożonej z funkcją prawdopodobieństwa: $\sum_{s \in S} p(s) \cdot f \circ p(s)$
  - wybranie jako tej funkcji $f = \log$
- Funkcja logarytmiczna odgrywa kluczowe znaczenie w naszej percepcji
- Prawo Webera-Fechnera: odbierana przez nasze zmysły percepcja (P) zmiany bodźca (S, od słowa *stimuli*) jest proporcjonalna nie do absolutnej, ale do względnej zmiany tego bodźca
- percepcja prawdopodobieństwa

Entropia

- $H_r(S) \geqslant 0$, równość zachodzi jedynie wtedy, gdy całe prawdopodobieństwo jest skupione w jednym punkcie
- Entropia jest zawsze ograniczona przez logarytm rozmiaru przestrzeni możliwości $H_r(S) \leqslant \log_r |S|$, równość ma miejsce wtedy i tylko wtedy, gdy $p(s) = \frac{1}{|S|}$ dla wszystkich $s \in S$

Minimalna długość kodu

- Dla danego kodu φ, średnią długość kodu definiujemy jako $L(\varphi) = \sum_{s \in S} p(s) \cdot |\varphi(s)|$
- Dla danego S i parametru $r > 1$ niech $L_r(S)$ będzie minimum ze wszystkich $L(\varphi)$ dla dowolnego kodu $\varphi : S \to \Sigma^*$, gdzie $|\Sigma| = r$. Zauważmy, że na mocy Twierdzenia McMillana wystarczy, że znajdziemy minimum dla wszystkich kodów bezprefiksowych.
- Dla dowolnej skończonej przestrzeni probabilistycznej S
  $H_r(S) \leqslant L_r(S)$
  i równość zachodzi wtedy i tylko wtedy, gdy wszystkie prawdopodobieństwa $p(s)$ są potęgami $\frac{1}{r}$

Algorytm Huffmana

(dla zbioru kodowanego S i alfabetu Σ)
Jeśli $|S| \leqslant |\Sigma|$, przypisz po prostu jakieś symbole obiektom z S. W przeciwnym wypadku:

1. Jeśli $|\Sigma| > 2$, w razie konieczności uzupełnij S symbolami o prawdopodobieństwie 0, tak aby $|S| \mod (|\Sigma| - 1) = 1$.
2. Wybierz $k = |\Sigma|$ symboli $t_1, \ldots, t_k \in S$ o minimalnych prawdopodobieństwach.
3. Uruchom rekurencyjnie ten algorytm dla zbioru $S' = S - \{t_1, \ldots, t_k\} + \{\#\}$ z prawdopodobieństwami $q_s = p_s$ dla $s \in S$ i $q_\# = p_{t_1} + \ldots + p_{t_k}$.
4. Mając dane drzewo $T_{S'}$ z poprzedniego punktu skonstruuj drzewo T_S w następujący sposób: Dodaj k synów do słowa $T_{S'}(\#)$ i oznacz ich jako $t_1, \ldots, t_k$.

Algorytm Huffmana

- Dla każdej litery stwórz drzewo złożone tylko z korzenia i ustaw drzewa w malejącym porządku prawdopodobieństwa p_i użycia danej litery.
- Dopóki istnieją przynajmniej 2 drzewa
    - z drzew t_1 i t_2 o najmniejszych prawdopodobieństwach $p_1 i p_2$ utwórz drzewo o korzeniu z prawdopodobieństwem $p_1 + p_2$ i mające t_1 jako lewe i t_2 jako prawe poddrzewo.
- Przypisz 0 każdej lewej krawędzi i 1 prawej krawędzi drzewa.
- Utwórz słowo kodu dla każdej litery przechodząc drzewo od korzenia do liścia i łącząc napotkane 0 i 1.

Kodowanie par

- Zmniejszenie długości kodu przez poszerzenie samego pojęcia kodu.
- Niech $S = \{s_1, s_2\}$ z $p(s_1) = \frac{3}{4}$, $p(s_2) = \frac{1}{4}$, $L_2(S) = 1$, $H_2(S) < 1$.
- Nie możemy zakodować wiadomości $\alpha \in \Sigma^*$ w postaci krótszej niż sama α.
- Wyobraźmy sobie jednak następujące kodowanie par: $$\begin{aligned} s_1 s_1 &\mapsto 0 \\ s_1 s_2 &\mapsto 10 \\ s_2 s_1 &\mapsto 110 \\ s_2 s_2 &\mapsto 111 \end{aligned}$$
- Zgodnie z definicją to jest kod dla S^2. Rozważmy $S^2 = S \times S$ jako produkt (probabilistyczny) przestrzeni, w którym $p\left(s_i, s_j\right) = p(s_i) \cdot p(s_j)$
- Średnia długość kodowania dwuznakowych bloków będzie wynosić $\left(\frac{3}{4}\right)^2 \cdot 1 + \frac{3}{4} \cdot \frac{1}{4} \cdot (2+3) + \left(\frac{1}{4}\right)^2 \cdot 3 = \frac{9}{16} + \frac{15}{16} + \frac{3}{16} = \frac{27}{16} < 2$
- Dla kodów złożonych z większej liczby znaków można otrzymać efektywniejsze kodowania.
- Nie można jednak zejść poniżej granicy entropii.

Entropia przestrzeni produktowej

- Niech S^n będzie n-tą potęgą przestrzeni S z prawdopodobieństwem $p(s_1, \ldots, s_n) = p(s_1) \cdot \ldots \cdot p(s_n)$.
- Wtedy
  $H_r S^n = n \cdot H_r S$

Adaptacyjna metoda Huffmana - algorytm kodowania dynamicznego

1. Utwórz początkowe drzewo (dwa liście: NEW, waga=0 oraz END, waga=1)
2. Jeśli jest to koniec źródła, zakończ cały proces, inaczej wczytaj znak ze źródła.
3. Sprawdź, czy znak ma już swoją reprezentację na drzewie przeszukując je od początku.
4. Gdy ma już swoją reprezentację, zmodyfikuj wagi począwszy od liścia reprezentującego nowy znak i odpowiednio zmodyfikuj drzewo (jeśli zajdzie potrzeba). Wyemituj do wyjścia kod tego węzła. Skocz do pkt. 2.
5. Gdy znak nie ma jeszcze swojej reprezentacji na drzewie, utwórz nowy węzeł o dwóch liściach: NEW, waga=0, oraz liścia reprezentującego nowy znak o wadze=1, nowy węzeł podłącz do starego liścia NEW na drzewie. Odpowiednio zmodyfikuj wagi na drzewie i wyemituj kod nowego liścia NEW oraz nowy znak. Idź do pkt. 2.

Algorytm Weavera-Hankamera

Podziel litery źródła $S = \{x_1, \ldots, x_n\}$, w którym każda litera x_i ma długość L bitów, na dwa zbiory,
$S_1 = \{x : p(x) > \frac{1}{2^L}\}$ i $S_2 = \{x : p(x) \leqslant \frac{1}{2^L}\}$;
$p(\mathrm{ELSE}) = \sum_{x \in S_2} p(x)$;
utwórz kod Huffmana dla zbioru $S_0 = S_1 \cup \{\mathrm{ELSE}\}$;
słowo kodu dla litery $x \in S_2$ jest złożeniem słowa kodu dla zbioru ELSE i litery x;

Minimalna długość kodu - kontynuacja

- Aby oszacować $\frac{L_r(S^n)}{n} - H_r(S)$, zaczniemy od uzupełnienia naszej nierówności o górne ograniczenie.
- Dla dowolnej skończonej przestrzeni probabilistycznej S i $r \geqslant 2$, istnieje kod $\varphi : S \to \Sigma^*(gdzie|\Sigma| = r)$, spełniający $L(\varphi) \leqslant H_r(S) + 1$
- Z tego wynika $H_r(S) \leqslant L_r(S) \leqslant H_r(S) + 1$
- Ścisła nierówność $L_r(S) < H_r(S) + 1$ jest prawdziwa za wyjątkiem przypadku $p(s) = 1$ dla pewnego $s \in S$ (wtedy $H_r(S) = 0$).

- **Pierwsze Twierdzenie Shannona**

  Dla każdej skończonej przestrzeni probabilistycznej S i $r \geqslant 2$

  $\lim_{n\to\infty} \frac{L_r(S^n)}{n} = H_r(S)$.

Kodowanie Shannona

Dane jest źródło $S = \{x_1, x_2, \ldots\}$ i stowarzyszone z nimi prawdopodobieństwa $p = \{p_1, p_2, \ldots\}$.

1. Prawdopodobieństwa (a wraz z nimi symbole) są sortowane w porządku nierosnącym, tj. $p_i \geqslant p_{i+1}$.
2. Następnie dla tak uporządkowanych danych oblicza się niepełne prawdopodobieństwo komulatywne: $P(x_i) = p_1 + p_2 + \ldots + p_{i-1}$ – jest to suma prawdopodobieństw elementów od 1 do $i - 1$.
3. Kodowanie Shannona polega na wzięciu $\lceil -\log_2 p_i \rceil$ (długość Shannona) pierwszych bitów binarnego rozwinięcia liczby P_i (brane są bity po przecinku).

Średnia długość kodów mieści się w przedziale $[H(S), H(S) + 1)$.

Algorytm Shannona-Fano

- Uporządkuj ciąg s nierosnąco wg prawdopodobieństw p_i
- Jeśli s zawiera 2 symbole,
  - do słowa kodu pierwszej litery dodaj 0, do słowa kodu drugiej litery dołącz 1.
- w przeciwnym razie, jeśli s zawiera więcej niż dwa symbole,
  - podziel go na dwa podciągi s_1 i s_2 tak, żeby różnica między sumą prawdopodobieństw liter z s_1 i s_2 była najmniejsza.
  - Do słów kodu symboli z s_1 dołącz 0, do kodów symboli z s_2 dołącz 1.
  - Wywołaj rekurencyjnie funkcje: Shannon-Fano(s_1) oraz Shannon-Fano(s_2).

Kodowanie LZ77

- Bufor słownikowy jest wypełniany pierwszym symbolem wejściowym i ten symbol jest zapisywany na wyjście.
- Do bufora wejściowego wstawiane jest n pierwszych symboli wejściowych.
- Dopóki w buforze wejściowym są jakieś dane:
  - W obrębie bufora słownikowego wyszukiwany jest najdłuższy podciąg równy początkowi bufora wejściowego (najdłuższy prefiks bufora kodowania). Wynikiem wyszukiwania jest indeks $P \in [0 \ldots k-1]$ początku wyszukanego podciągu oraz jego długość C, ograniczona z góry przez $n-1$. Część podciągu może być wspólna z buforem wejściowym! Jeśli podciągu nie uda się znaleźć, to P może mieć dowolną wartość, natomiast $C = 0$.
  - Na wyjście wyprowadzana jest trójka (P, C, symbol S z bufora wejściowego następujący po dopasowanym podciągu).
  - Okno (bufor słownikowy + bufor wejściowy) przesuwane jest w lewo o $C+1$ pozycji i na koniec bufora dopisywane jest tyleż kolejnych symboli wejściowych (o ile jeszcze są jakieś).

Dekodowanie LZ77

- Bufor słownikowy jest wypełniany początkowym symbolem
- Dla wszystkich trójek (P,C,S) wykonuj:
  - Skopiuj z bufora słownikowego do bufora wyjściowego symbole z zakresu $P \ldots P + C - 1$.
  - Doklej na koniec bufora wyjściowego symbol S
  - Na wyjście wyprowadź $C + 1$ początkowych symboli z bufora wejściowego.
  - Przesuń zawartość bufora o $C + 1$ pozycji w lewo.

Kodowanie LZ78

Kompresowany jest ciąg S zawierający n symboli.

- Wyczyść słownik.
- $i := 0$ (i – indeks pierwszego, nieprzetworzonego symbolu w S).
- Dopóki $i < n$, wykonuj:
  - Wyszukaj w słowniku najdłuższy podciąg równy początkowi nieprzetworzonych jeszcze symboli (podciąg $S[i \ldots]$).
  - Jeśli udało się znaleźć taki podciąg, to wynikiem wyszukiwania jest jego indeks k w słowniku; dodatkowo słowo wskazywane przez ten indeks ma pewną długość m. Na wyjście wypisz parę (indeks, pierwszy niedopasowany symbol), czyli $(k, S[i+m])$ oraz dodaj do słownika znaleziony podciąg przedłużony o symbol $S[i+m]$ (innymi słowy podciąg $S[i \ldots i+m]$). Zwiększ $i := i + m$.
  - Jeśli nie udało się znaleźć żadnego podciągu, to znaczy, że w słowniku nie ma jeszcze symbolu $S[i]$. Wówczas do słownika dodawany jest ten symbol, a na wyjście wypisywana para $(0, S[i])$. Indeks 0 jest tutaj umowny, w ogólnym przypadku chodzi o jakąś wyróżnioną liczbę. Zwiększ i o jeden.

Dekodowanie LZ78

- Wyczyść słownik.
- Dla wszystkich par (indeks, symbol – ozn. k, s) wykonuj:
  - Jeśli $k = 0$, dodaj symbol s do słownika. Na wyjście wypisz symbol s.
  - Jeśli $k > 0$, weź ze słownika słowo w spod indeksu k. Na wyjście wypisz słowo w oraz symbol s. Do słownika pod kolejnym indeksem dodaj słowo $w + s$.

Kodowanie LZSS

- Wypełnij słownik pierwszym symbolem, wypisz ten symbol na wyjście; wypełnij bufor wejściowy n pierwszymi symbolami wejściowymi.
- Dopóki w buforze wejściowym są dane:
  - Wyszukaj w słowniku najdłuższy podciąg równy początkowi bufora wejściowego - wynikiem są liczby P i C.
  - Jeśli rozmiar pary (P, C) jest mniejszy od rozmiaru znalezionego podciągu, zapisz na wyjście trójkę $(0,P,C)$, przesuń cały bufor o C pozycji w lewo i wprowadź do bufora wejściowego tyle samo kolejnych symboli.
  - W przeciwnym razie wypisz na wyjście parę $(1, S')$, przesuń cały bufor o 1 pozycję w lewo i wprowadź do bufora wejściowego kolejny symbol wejściowy.

Dekodowanie LZSS

- Wypełnij słownik pierwszym symbolem.
- Dla kolejnych danych (par i trójek) powtarzaj:
  - Jeśli mamy do czynienia z trójką $(0,P,C)$ to skopiuj ze słownika na początek bufora wyjściowego symbole z zakresu $P \ldots P+C-1$, wypisz na wyjście skopiowane symbole i przesuń cały bufor o C pozycji w lewo.
  - Jeśli mamy do czynienia z dwójką $(1,S)$ to skopiuj symbol S na początek bufora wyjściowego, wypisz na wyjście ten symbol i przesuń cały bufor o jedną pozycję w lewo.

Kodowanie LZW

- Wypełnij słownik alfabetem źródła informacji.
- $c :=$ pierwszy symbol wejściowy
- Dopóki są dane na wejściu:
  - Wczytaj znak s.
  - Jeżeli ciąg $c + s$ znajduje się w słowniku, przedłuż ciąg c, tj. $c := c + s$
  - Jeśli ciągu $c + s$ nie ma w słowniku, wówczas:
    - wypisz kod dla c (c znajduje się w słowniku)
    - dodaj ciąg $c + s$ do słownika
    - przypisz $c := s$.
- Na końcu wypisz na wyjście kod związany c.

Dekodowanie LZW

- Wypełnij słownik alfabetem źródła informacji.
- $pk :=$ pierwszy kod skompresowanych danych
- Wypisz na wyjście ciąg związany z kodem pk, tj. słownik[pk]
- Dopóki są jeszcze jakieś słowa kodu:
  - Wczytaj kod k
  - $pc :=$ słownik[pk] – ciąg skojarzony z poprzednim kodem
  - Jeśli słowo k jest w słowniku, dodaj do słownika ciąg ($pc +$ pierwszy symbol ciągu słownik[k]), a na wyjście wypisz cały ciąg słownik[k].
  - W przeciwnym razie (przypadek *scscs*) dodaj do słownika ciąg ($pc +$ pierwszy symbol pc) i tenże ciąg wypisz na wyjście.
  - $pk := k$

Kodowanie arytmetyczne

Dany jest zbiór symboli $S = \{x_1, x_2, \ldots\}$ oraz stowarzyszony z nim zbiór prawdopodobieństw $p = \{p_1, p_2, \ldots\}$. Jeden z symboli jest wyróżniony – jego wystąpienie oznacza koniec komunikatu, co zapobiega wystąpieniu niejednoznaczności; ewentualnie zamiast wprowadzenia dodatkowego symbolu można przesyłać długość kodowanego ciągu.
Na początku dany jest przedział $P = [0, 1)$, który dzielony jest na podprzedziały o szerokościach równych kolejnym prawdopodobieństwom p_i, czyli otrzymywany jest ciąg podprzedziałów
$[0, p_1), [p_1, p_1 + p_2), [p_1 + p_2, p_1 + p_2 + p_3), [p_1 + p_2 + p_3, p_1 + p_2 + p_3 + p_4), \ldots$.
Kolejnym podprzedziałom (ozn. R_i) odpowiadają symbole ze zbioru S.

- Dla kolejnych symboli c:
  - Określ, który podprzedział bieżącego przedziału P odpowiada literze c – wynikiem jest R_i.
  - Weź nowy przedział $P := R_i$ – następuje zawężenie przedziału
  - Podziel ten przedział P na podprzedziały w sposób analogiczny do tego, jak to miało miejsce na samym początku (chodzi o zachowanie proporcji szerokości podprzedziałów).
- Zwróć liczbę jednoznacznie wskazującą przedział P (najczęściej dolne ograniczenie, albo średnia dolnego i górnego ograniczenia).

Dekodowanie arytmetyczne

- x – liczba (kod)
- $P = [0, 1)$
- Wykonuj w pętli:
    - Podziel P na podprzedziały R_i
    - Znajdź podprzedział R_i, do którego należy x.
    - $P := R_i$
    - Wypisz i-ty symbol na wyjście
    - Jeśli i-ty symbol był symbolem końcowym, zakończ pętlę