Asymetryczne systemy liczbowe – wygodna praca z ułamkowymi bitami

*Jarosław DUDA**

* Autor jest adiunktem w Instytucie Informatyki i Matematyki Komputerowej na Uniwersytecie Jagiellońskim, a zarazem autorem ogólnej koncepcji ANS oraz wszystkich opisywanych w tekście jej wariantów.

Systemy liczbowe mają długą historię – zaczynając od liczb egipskich, chińskich, babilońskich, rzymskich i in. aż do współczesnej dominacji systemu dziesiątkowego dla ludzi oraz dwójkowego dla komputerów. To jeszcze nie koniec – w ostatnich latach dane w naszych komputerach coraz częściej zapisane są asymetrycznymi systemami liczbowymi: ANS, od ang. *Asymmetric Numeral Systems*.

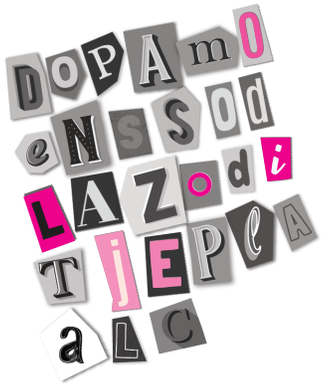
Jest tak, ponieważ system dwójkowy jest idealny tylko dla zapisu ciągu nieskorelowanych 0 i 1 o jednakowym prawdopodobieństwie $\Pr(0) = \Pr(1) = 1/2$, gdyż takie zdarzenia rzeczywiście niosą po jednym bicie informacji. Natomiast w rzeczywistości operujemy na zdarzeniach o bardziej skomplikowanych rozkładach, statystykach, często niosących ułamkowe ilości bitów informacji, np. chociażby cyfra dziesiętna niesie $\lg(10) \approx 3{,}32$ bitów lub mniej ($\lg \equiv \log_2$). W takich przypadkach bezpośrednie zapisanie danych w systemie dwójkowym może okazać się (w sensie rozmiaru danych) dalece nieoptymalne. ANS próbuje więc uogólniać systemy liczbowe, optymalizując je do dowolnego wybranego rozkładu prawdopodobieństwa symboli reprezentujących zdarzenia. Dzięki temu ANS w ostatnich latach stał się podstawowym sposobem pracy na ułamkowych bitach w zapisie danych, które przechowujemy i przesyłamy – jako prostszy zastępuje kodowanie arytmetyczne (`pl.wikipedia.org/wiki/Kodowanie_arytmetyczne`), pozwalając na szybsze implementacje, nawet do 30 razy.

$M = 2^N$ różnych wartości wymaga $N = \lg M$ bitów do jednoznacznego zapisu. Mówimy więc, że jednostajne zdarzenie o n wartościach *niesie* $\lg(n)$ bitów informacji.

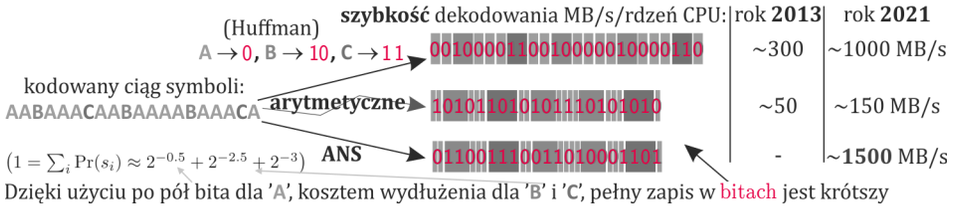
Zawartość informacyjną zwykle mierzymy entropią Shannona, którą chyba najłatwiej zrozumieć asymptotycznym zachowaniem dwumianu Newtona w następującym przypadku: rozważamy ciągi zero-jedynkowe o długości n, z których dokładnie p procent pozycji stanowią jedynki. Oszacujmy liczbę takich ciągów. Przybliżenie wzorem Stirlinga $n! \approx (n/e)^n$ daje nam:

$$\binom{n}{pn} = \frac{n!}{(pn)!\,((1-p)n)!} \approx \frac{(n/e)^n}{(pn/e)^{pn}\,((1-p)n/e)^{(1-p)n}} = \\ = 2^{n(-p\lg(p)-(1-p)\lg(1-p))} = 2^{nh(p)},$$

przy czym $h(p) := -p\lg(p) - (1-p)\lg(1-p)$ oznacza entropię Shannona. Aby optymalnie (możliwie krótko) opisać jedną z tych kombinacji, potrzebujemy około $\lg\binom{n}{pn} \approx nh(p)$ bitów informacji – gdyż ogólnie K bitów pozwala *zakodować* jedną z 2^K różnych możliwości. Otrzymany wynik pokazuje istotną różnicę między prostym zapisem kolejnych cyfr za pomocą kolejnych bitów (jeden znak – jeden bit) a takim, w którym bierzemy pod uwagę rozkład prawdopodobieństwa. W naszym przypadku – kodowanie trywialne dałoby n bitów, a takie uwzględniające liczbę jedynek – około $nh(p)$ bitów, czyli średnio około $h(p)$ bita na jeden znak ($h(0) = h(1) = 0$, $h(0{,}11) = h(0{,}89) \approx 0{,}5$, maksimum $h(1/2) = 1$).



Interpretując entropię Shannona jako średnią ważoną, z powyższych rozważań dostajemy ogólną praktyczną regułę: **symbol o prawdopodobieństwie** p **niesie** $\lg(1/p)$ **bitów informacji**, czyli na przykład 1 bit dla $p = 1/2$, 3 bity dla $p = 1/8$, ale może być też ułamkowa ilość, np. pół bita dla $p = 2^{-0{,}5} \approx 0{,}707$.



Rys. 1. Wizualizacja skrócenia zapisu dzięki uwzględnieniu ułamkowych bitów

Rozwiązanie zadania F 1035.
Napięcie między chmurą i powierzchnią ziemi w momencie rozpoczęcia wyładowania wynosi $U = h \cdot E = 1 \cdot 10^8$ V. W celu oszacowania zgromadzonego ładunku potraktujmy układ chmura–ziemia jak kondensator płaski (jak zwykle, pomijamy efekty brzegowe) o pojemności

$$C = \frac{\varepsilon_0 S}{h},$$

przy czym $S = \pi D^2/4$ jest powierzchnią podstawy chmury. Ładunek zgromadzony na kondensatorze pod napięciem U wynosi $Q = C \cdot U$, a energia $W = CU^2/2$. Po podstawieniu danych liczbowych otrzymujemy oszacowanie: $Q \approx 280$ C, $W \approx 1{,}4 \cdot 10^{10}$ J. Rozładowanie takiego kondensatora podczas burzy następuje w kilkunastu do kilkudziesięciu wyładowaniach – przyjmijmy, że wyładowań jest 20. Otrzymujemy, że w pojedynczej błyskawicy przepływa ładunek około 14 C i wyzwalana jest energia rzędu $7{\cdot}10^8$ J.
Otrzymane tu oszacowania są zbliżone do wyników pomiarów: C. R. Maggio, T. C. Marshall, M. Stolzenburg, *Journal of Geophysical Research* **114**, D14203 (2009).

Rozwiązanie zadania F 1036.
Kończyny muszą unieść ciężar ssaka $F = mg$, przy czym g jest przyspieszeniem ziemskim. Skoro kształty ssaków są podobne i gatunki różnią się głównie wielkością, to ciśnienie działające na kości kończyn jest proporcjonalne do masy m i odwrotnie proporcjonalne do pola przekroju kończyny proporcjonalnego do kwadratu jej grubości d^2. Wynika stąd, że $pd^2 \propto mg$, przy czym p oznacza wytrzymałość kości na ściskanie, a więc:

$$d \propto \sqrt{\frac{mg}{p}}.$$

Jeśli za miarę wielkości zwierzęcia przyjmiemy jego „rozmiar liniowy" L, to jego masa byłaby proporcjonalna do L^3, $m \propto L^3$, a masa kończyn $m_{nogi} \propto Ld^2$, czyli $m_{nogi} \propto m^{1/3} m = m^{4/3}$. Pomiary wykazują, że masa szkieletu (i kończyn) rośnie z masą ssaka, ale z wykładnikiem około 1,09, a nie 1,33, jak wynika to z naszej analizy. Można stąd wyciągnąć wniosek, że w przebiegu ewolucji wytrzymałość kości na ściskanie nie decyduje o rozmiarach zwierząt.
Knut Schmidt-Nielsen, *Dlaczego tak ważne są rozmiary zwierząt. Skalowanie*, Wydawnictwo Naukowe PWN, Warszawa 1994.

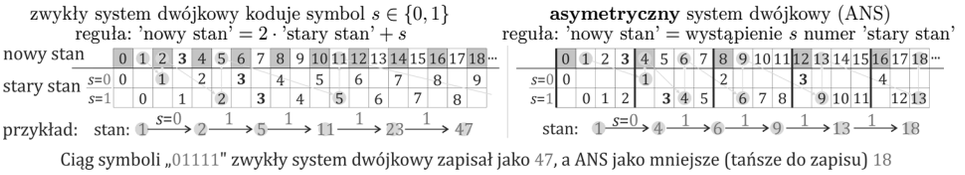
Nie wszystkie popularne systemy kodowania (kompresji) tekstów w sposób optymalny eksploatują to zjawisko. Rysunek 1 wizualizuje porównanie zachowania kilku algorytmów dla rozkładu literek $\Pr(A) \approx 2^{-0,5}$, $\Pr(B) = 2^{-2,5}$, $\Pr(C) = 2^{-3}$.

Na pewnym poziomie abstrakcji możemy sobie wyobrażać, że kodowanie polega na zapisaniu całej informacji w jednej bardzo dużej liczbie, którą będziemy oznaczać jako $x \in \mathbb{N}$. Na pytanie, ile niesie ona bitów informacji, chciałoby się ją zlogarytmować, po czym zaokrąglić do liczby naturalnej. Tutaj jednak chcemy uwzględnić ułamkowe bity, więc uciąglamy naszą odpowiedź i wyobrażamy sobie, że **liczba x zawiera dokładnie $\lg(x)$ bitów informacji**. Kluczowym pytaniem jest: co zrobić, gdy chcemy dołożyć do x nową informację na temat kolejnego symbolu s, o prawdopodobieństwie $\Pr(s) = p$. Wiedząc, że niesie on $\lg(1/p)$ bitów, wnioskujemy, że nowy stan $x' = C(x,s)$ zawierający informacje z obu powinien nieść: $\lg(x') \approx \lg(x) + \lg(1/p)$ bitów, czyli $x' = C(x,s) \approx x/\Pr(s)$. Im lepsze to przybliżenie, tym krótszy dostaniemy zapis. Jak to wygląda w różnych znanych algorytmach?

Wracając do systemu dwójkowego, takie dołożenie nowej informacji z $s \in \{0,1\}$ do informacji już zawartej w liczbie $x \in \mathbb{N}$ moglibyśmy wykonać poprzez dostawienie s jako najstarszej cyfry zapisu: $x' = x + s\,2^k$, przy czym k to pierwsza wolna pozycja. Tutaj s wybiera jeden z dużych przedziałów dla x': $[0, 2^k)$ albo $[2^k, 2^{k+1})$. Kodowanie arytmetyczne optymalizuje to podejście dla dowolnego rozkładu s poprzez zmianę długości tych dwóch podprzedziałów – tak żeby ich proporcje zgadzały się z założonym rozkładem prawdopodobieństwa. Wymaga to pamiętania na przykład obu krańców aktualnego przedziału, ciągłego ich przetwarzania – co odpowiada dodatkowej kontroli pozycji k w $x' = x + s\,2^k$.

ANS upraszcza ten pomysł dzięki idei umieszczania nowej informacji z symbolu s na najmłodszej pozycji x, jednocześnie przesuwając pozostałe bity: $C(x,s) = x' = 2x + s$. Już nie wymaga to kontroli pozycji k – możemy pracować na pojedynczej liczbie, a to prowadzi do szybszych implementacji. Tutaj funkcja dekodująca to $D(x') = (\lfloor x/2 \rfloor, \mathrm{mod}(x,2))$, co pozwala jednoznacznie odwrócić taki proces: $C(D(x')) = x'$, $D(C(x,s)) = (x,s)$.

Regułę $C(x,s) = x' = 2x + s$ możemy zapisać jako „x przechodzi w $C(x,s)$ jako x-te wystąpienie liczby parzystej (I_0) lub nieparzystej (I_1)". Uasymetryczniamy ją w ANS, po prostu przedefiniowując ten rozłączny podział: $\mathbb{N} = \sqcup_s I_s$, tym razem tak, że rozmiar $|I_s \cap [0,x)| \approx p_s\, x$ dla dowolnego x i ustalonego $\Pr(s) = p_s \in (0,1)$, już niekoniecznie 1/2. Czyli wybieramy w miarę jednorodne rozrzucenie symboli – funkcję $\overline{s} : \mathbb{N} \to \mathcal{A}$ dla alfabetu $\mathcal{A}$, uogólniającą podział na liczby parzyste/nieparzyste, po czym $I_s = \{x : \overline{s}(x) = s\}$. Dekodowanie z x' wymaga najpierw znalezienia symbolu s takiego, że $x' \in I_s$, po czym odczytania, którym jest tam wystąpieniem. Dla zgrubnego wyobrażenia sobie, jak to działa, można zerknąć na rysunek 2, przy czym $I_0 = \{x : \mathrm{mod}(x,4) = 0\}$, $I_1 = \{x : \mathrm{mod}(x,4) \in \{1,2,3\}\}$.



Rys. 2. Przykład porównania zwykłego systemu dwójkowego i asymetrycznego

W praktyce trzeba zautomatyzować ten proces, na co jest kilka sposobów. Zacznijmy od bardziej dydaktycznego wariantu jednorodnego **uABS** (z 2006 r., *uniform, binary*). Dla dwuelementowego alfabetu: $\Pr(1) = p \in (0,1)$, $\Pr(0) = 1 - p$, liczba wystąpień '1' do pozycji x powinna być mniej więcej $x \cdot p$, więc zaokrąglamy na przykład w górę: mówimy, że na pozycjach $[0,x)$ wystąpiło $\lceil x \cdot p \rceil$ razy, czyli '1' jest tam, gdzie $\lceil x \cdot p \rceil$ wzrasta, w pozostałych jest '0'. Daje to funkcję dekodującą D, dla której łatwo znaleźć odwrotność C:

Rozwiązanie zadania M 1690.
Oznaczmy punkty jako $A_1, A_2, \ldots, A_{3n}$. Ze względu na to, że tych punktów jest skończenie wiele, możemy wybrać prostą ℓ taką, że rzuty prostokątne dowolnych dwóch punktów nie pokrywają się. Przez X_i oznaczmy rzut prostokątny punktu A_i na ℓ. Bez szkody dla ogólności przyjmijmy, że rzuty prostokątne leżą na prostej ℓ w kolejności: $X_1, X_2, \ldots, X_{3n}$. Wówczas trójkąty $A_iA_{i+1}A_{i+2}$ dla $i = 1, 4, 7, \ldots, 3n-2$ są parami rozłączne.

Rozwiązanie zadania M 1691.
Rozpatrzmy taki punkt O leżący wewnątrz trójkąta $A'B'C'$, że

$$\sphericalangle A'OB' = \sphericalangle B'OC' = \sphericalangle C'OA' = 120^\circ$$

oraz x, y, z są długościami odcinków OA', OB', OC', odpowiednio.
Z twierdzenia cosinusów dla trójkątów $A'OB'$, $B'OC'$ oraz $C'OA'$ wynikają następujące równości:

$$\begin{cases} x^2 + xy + y^2 = A'B'^2 \\ y^2 + yz + z^2 = B'C'^2 \\ z^2 + zx + x^2 = C'A'^2 \end{cases}$$

Z układu danego w treści zadania wynika, że trójkąty ABC i $A'B'C'$ są przystające!
Stosując teraz twierdzenie sinusów widzimy, że

$$[ABC] = [AOB] + [BOC] + [COA] = \\ = (xy + yz + zx)\frac{\sqrt{3}}{4}.$$

Wobec tego na podstawie twierdzenia Herona dostajemy

$$xy + yz + zx = \\ = \frac{4\sqrt{3}}{3}\sqrt{s(s-a)(s-b)(s-c)},$$

gdzie $s = \frac{a+b+c}{2}$ jest połową obwodu trójkąta.

Rozwiązanie zadania M 1692.
Zauważmy, że

$$(1) \qquad \frac{1+bc}{b-c}\cdot\frac{1+ab}{a-b} + \frac{1+ca}{c-a}\cdot\frac{1+ab}{a-b} + \\ + \frac{1+bc}{b-c}\cdot\frac{1+ca}{c-a} = 1,$$

stąd jeśli

$$1 < d = \mathrm{NWD}\left(\frac{1+bc}{b-c}, \frac{1+ca}{c-a}, \frac{1+ab}{a-b}\right),$$

to d dzieli lewą stronę (1), czyli $d \mid 1$ – sprzeczność.
Jak można otrzymać równość (1)? Zauważmy, że dla pewnych x, y, $z \in (0, \pi/2)$ zachodzą równości $a = \mathrm{tg}(x)$, $b = \mathrm{tg}(y)$ i $c = \mathrm{tg}(z)$. Wówczas

$$\frac{a-b}{1+ab} = \frac{\mathrm{tg}(x) - \mathrm{tg}(y)}{1 + \mathrm{tg}(x)\,\mathrm{tg}(y)} = \mathrm{tg}(x-y).$$

Podobnie

$$\frac{b-c}{1+bc} = \mathrm{tg}(y-z), \quad \frac{c-a}{1+ac} = \mathrm{tg}(z-x).$$

Ponieważ $(x-y) + (y-z) + (z-x) = 0$, to ze wzoru na tangens sumy wynika, że

$$\mathrm{tg}(x-y) + \mathrm{tg}(y-z) + \mathrm{tg}(z-x) = \\ = \mathrm{tg}(x-y)\,\mathrm{tg}(y-z)\,\mathrm{tg}(z-x),$$

czyli (1).

$$D(x) = (x_s, s): \quad s = \lceil (x+1)p \rceil - \lceil xp \rceil, \quad x_1 = \lceil xp \rceil, \quad x_0 = x - x_1,$$
$$C(x,0) = \lceil (x+1)/(1-p) \rceil - 1, \qquad C(x,1) = \lfloor x/p \rfloor.$$

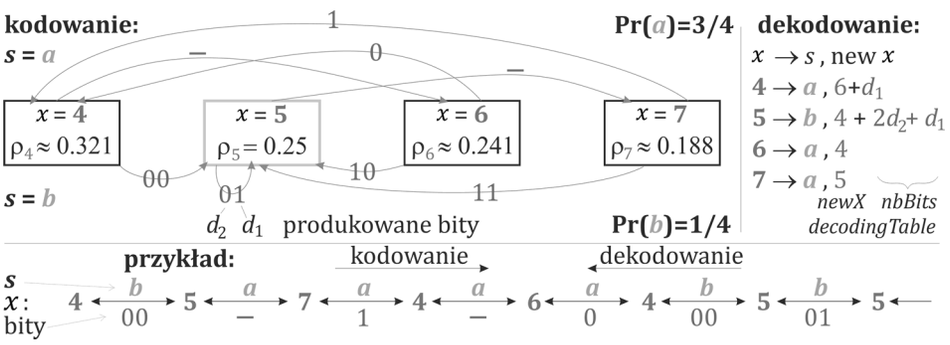
Wygodniej jest pracować na alfabecie większym niż dwójkowy, ale uogólnienie powyższego podejścia pozostaje problemem otwartym. Na szczęście w 2013 roku powstał powszechnie dziś używany wariant przedziałowy **rANS** – gorzej przybliżający $C(x,s) \approx x/\Pr(s)$, ale wciąż wystarczająco dobrze dla dużych x. Dzielimy w nim $\mathbb{N}$ na przedziały wielkości 2^n, każdy z nich w identyczny sposób na podprzedziały odpowiadające symbolom, zgodnie z założonym rozkładem prawdopodobieństwa. Czyli symbol s ma tam $f_s \approx 2^n \Pr(s)$ wystąpień: pozycje $[c_s, c_{s+1})$ dla $c_s = f_0 + \ldots + f_{s-1}$. Na przykład dla $2^n = 4$, $f_0 = 1$, $f_1 = 3$ (patrz prawy diagram rysunku 2). Dekodując x, korzystamy z tego, że jest on w $\lfloor x/2^n \rfloor$ dużym przedziale. Pozycja w tym przedziale to $\mathrm{mod}(x, 2^n)$, trzeba sprawdzić, któremu symbolowi odpowiada: $s = \overline{s}(\mathrm{mod}(x, 2^n))$ takie, że $c_s \leqslant \mathrm{mod}(x, 2^n) < c_{s+1}$. Potem odpowiednio przesuwając, dostajemy funkcję dekodującą i kodującą:

$$s = \overline{s}(\mathrm{mod}(x, 2^n)), \qquad D(x) = (f_s\lfloor x/2^n \rfloor + \mathrm{mod}(x, 2^n) - c_s, s),$$
$$C(x,s) = 2^n\lfloor x/f_s \rfloor + \mathrm{mod}(x, f_s) + c_s.$$

Dzięki użyciu szerokości 2^n część z tych operacji zamienia się m.in. w przesunięcia bitowe. Moglibyśmy zakodować ciąg symboli, używając powyższych wzorów uABS lub rANS, i na końcu np. zapisać dwójkowo ostateczne x. Ale rośnie ono bardzo szybko: $\lg(x) \approx$ entropia sekwencji, więc w praktyce dodaje się zwykle tzw. **renormalizację**: pilnowanie, żeby $x \in I$ dla I będącego przedziałem np. $I = [2^{16}, 2^{32})$. Jeśli dokodowując nowy symbol, wyszlibyśmy z tego przedziału, to wcześniej *zrzucamy* najmłodsze 16 bitów: $\mathrm{mod}(x, 2^{16})$ na strumień wyjściowy, po czym zmniejszamy stan $x \to \lfloor x/2^{16} \rfloor$ i kodujemy dalej. Ten proces łatwo można odwrócić: dekoder dla $x < 2^{16}$ przesuwa $x \to 2^{16}\,x$ oraz doczytuje ze strumienia najmłodsze 16 bitów.

Takie zrzucanie zakumulowanych pełnych bloków na strumień wyjściowy pozwala też po prostu stablicować całe zachowanie (wariant **tANS** z 2007 r.) na mniejszym przedziale, np. $I = [4, 8)$ na rysunku 3, w praktyce np. $I = [2048, 4096)$ dla alfabetu o wielkości 256, czyli pracy na bajtach. Dostajemy automat skończony (już niewymagający mnożenia/dzielenia), zoptymalizowany dla wybranego rozkładu prawdopodobieństwa. Dodatkowo mamy olbrzymią swobodę wyboru w miarę jednorodnego rozrzucenia symboli – daje to dodatkowe specyficzne zastosowanie, gdy dane chcemy nie tylko zwięźle zapisać, ale i zaszyfrować: możemy wówczas wyżej opisany proces przeprowadzić zależnie od klucza kryptograficznego, przy okazji za darmo szyfrując tak zakodowane dane.

Współcześnie nasze dane są coraz częściej zapisywane w omawianym wariancie tANS. Tak jest chociażby w produktach iPhone czy Mac firmy Apple (domyślny kompresor LZFSE), w kompresorze Facebook Zstandard zastępującym podstawowy zip m.in. w jądrze systemu Linux czy produktach wielu firm, jak Amazon czy IBM, też w Oodle Kraken popularnym w grach komputerowych. Jest on dobry dla stałych rozkładów, w bardziej skomplikowanych sytuacjach jest używany rANS – m.in. w kompresji DNA (CRAM), Google Draco dla danych 3D, JPEG XL, którym niedługo będą zapisane nasze zdjęcia i grafiki.



Rys. 3. Przykład budowy i użycia automatu tANS dla $I = [4, 8)$