Kody korekcyjne

Artur JEŻ

Każdemu zdarza się pomylić podczas pisania na klawiaturze. Nie każdy ciąg liter jest jednak sensownym słowem i dzięki tej nadmiarowości zwykle da się odtworzyć oryginalne słowo. Sytuacja staje się dużo trudniejsza przy wszelkiego rodzaju danych przetwarzanych przez komputery: numery kont bankowych, zawartości plików, transmisja radiowa... W jaki sposób wykrywać i poprawiać w nich błędy? Poniżej przedstawimy kody korekcyjne, które dodają do danych odpowiednią „nadmiarowość" i dzięki temu pozwalają wykrywać i poprawiać błędy.

Skupmy się na razie na ciągach bitów. Każdy zapewne słyszał o bicie kontroli parzystości (kod parzystości): ustalamy pewną wartość k (powiedzmy $k = 7$) i po każdych k bitach dodajemy ich sumę modulo 2. Przy odczytywaniu sprawdzamy, czy suma (modulo 2) kolejnych $k+1$ bitów wynosi 0: jeśli nie, to nastąpił błąd (i np. prosimy o powtórzenie). Innym naturalnym sposobem korekcji jest kod powtórzeniowy: każdy bit powtarzamy trzykrotnie i przy odczytywaniu „głosujemy większością". Mimo użycia trzykrotnie więcej znaków już dwa błędy wystarczą, by źle „poprawić" wiadomość: `010` może oznaczać zarówno `000` z jednym błędem, jak i `111` z dwoma błędami; intuicyjnie nie jest to więc dobre rozwiązanie. Innym przykładem jest kod Hamminga, który pozwala poprawić 1 błąd: dla bitów b_1, b_2, b_3, b_4 dopisujemy bity $b_1 + b_2 + b_4, b_1 + b_3 + b_4, b_2 + b_3 + b_4$. Cierpliwy Czytelnik może sprawdzić, że kod Hamminga faktycznie pozwala poprawić jeden błąd. Do takiego kodu można dodać bit parzystości, poprawia on wtedy dalej jeden błąd, ale wykrywa dwa błędy.

Podobne rozwiązanie jak kontrola bitu parzystości stosowane jest przy numerach rachunków bankowych (dwie wiodące cyfry kontrolne) czy numerach ISBN książek (ostatnia cyfra).

Richard Hamming opracował swój kod oraz jego uogólnienia w ramach walki z plagą błędów powstających przy wpisywaniu bitów programów komputerowych. Tak, kiedyś robiono to ręcznie.

Kod Hamminga z bitem parzystości znany jest też jako SECDED *Single Error Correction Double Error Detection* i jest powszechnie implementowany w systemach komputerowych.

Podejdźmy bardziej systematycznie do konstrukcji kodów korekcyjnych. Ustalmy zbiór $\mathbb{F}$ symboli, np. $\mathbb{F} = \{0, 1\}$. Kodem będzie dla nas pewien zbiór $C \subseteq \mathbb{F}^n$ ciągów n-elementowych, które traktujemy jako wektory, nazywamy je też słowami kodowymi; zwykle C jest podprzestrzenią liniową i dlatego $|C| = |\mathbb{F}|^k$ dla pewnego $k \leqslant n$. Jest tak np. dla wszystkich rozważanych poprzednio kodów. Przypuśćmy, że otrzymaliśmy wiadomość $\vec{w} = (w_1, \ldots, w_n) \notin C$ spoza C. Do jakiego słowa kodowego należy ją poprawić? Naturalne jest poprawienie $\vec{w}$ do $\vec{c} \in C$, które „różni się najmniej". Formalnie odległość Hamminga $d_H(\vec{w}, \vec{c})$ między $\vec{w}, \vec{c} \in \mathbb{F}^n$ to liczba pozycji, na których $\vec{w}$ i $\vec{c}$ się różnią; później przyda nam się, że d_H spełnia nierówność trójkąta, tj. $d_H(\vec{u}, \vec{v}) \leqslant d_H(\vec{u}, \vec{w}) + d_H(\vec{w}, \vec{v})$. Reasumując, $\vec{w}$ poprawiamy do $\vec{c} \in C$ o minimalnej odległości Hamminga od $\vec{w}$.

Jest subtelna różnica między kodem, czyli zbiorem ciągów, a kodowaniem, czyli przekształcaniem oryginalnej wiadomości w słowa kodowe. Najczęściej jednak kodowanie jest naturalne, a kod jest definiowany jako jego obraz, dlatego nie będziemy się nad tym rozwodzić.

Poprawianie, które „różni się najmniej", jest naturalne, ale jego poprawność zależy od modelu: tu zakładamy, że błędy są (w odpowiednim sensie) losowe, jednak w pewnych scenariuszach należy poprawiać inaczej: np. możemy wiedzieć, że `1` jest zawsze przekazane poprawnie, a `0` niekoniecznie.

Przedstawimy teraz rodzinę kodów korekcyjnych Reeda–Solomona, która dla zadanych $k \leqslant n$ pozwala na poprawienie $\lfloor (n-k)/2 \rfloor$ błędów. Wiadomość $\vec{m} = (m_0, \ldots, m_{k-1})$ kodujemy jako wartość wielomianu $w(x) = \sum_{i=0}^{k-1} m_i x^i$ w punktach $0, 1, \ldots, n-1$ (można użyć innych punktów, dla ustalenia uwagi używać jednak będziemy $0, 1, \ldots, n-1$), to znaczy jako $(w(0), \ldots, w(n-1))$. Odległość dwóch różnych słów kodowych to przynajmniej $n-k+1$: jeśli $\vec{w} = (w_0, \ldots, w_{n-1})$ i $\vec{v} = (v_0, \ldots, v_{n-1})$ są słowami kodowymi, to $\vec{w} - \vec{v}$ to ciąg wartości wielomianu $w(x) - v(x)$ w punktach $0, \ldots, n-1$. Wielomian $w(x) - v(x)$ jest niezerowy i $\deg(w(x) - v(x)) < k$, ma więc najwyżej $k-1$ miejsc zerowych. Tak więc przynajmniej $n-k+1$ spośród $(w_0 - v_0, \ldots, w_{n-1} - v_{n-1})$ to niezerowe liczby, co oznacza, że $d_H(\vec{w}, \vec{v}) \geqslant n-k+1$. Jeśli przy przesyłaniu kodu $\vec{w} \in C$ otrzymamy $\vec{u}$ przy nie więcej niż $\lfloor (n-k)/2 \rfloor$ błędach, czyli $d_H(\vec{w}, \vec{u}) \leqslant \lfloor (n-k)/2 \rfloor$, to faktycznie $\vec{w}$ jest najbliższym elementem z C dla $\vec{u}$: jeśli najbliższym byłby $\vec{v} \in C$, to wtedy nierówności

$$d_H(\vec{v}, \vec{u}) \leqslant d_H(\vec{w}, \vec{u}) \leqslant \lfloor (n-k)/2 \rfloor \text{ i } d_H(\vec{v}, \vec{w}) \leqslant d_H(\vec{v}, \vec{u}) + d_H(\vec{u}, \vec{w})$$

dają $d_H(\vec{v}, \vec{w}) \leqslant n-k$, sprzeczność.

Korekcja Reeda–Solomona jest optymalna: kod może poprawiać najwyżej $\lfloor (n-k)/2 \rfloor$ błędów; jest to tzw. ograniczenie Singletona.

Jak na razie nie powiedzieliśmy nic o współczynnikach wielomianów. Używanie liczb rzeczywistych jest problematyczne: liczby w kodowaniu $w_0, \ldots, w_{n-1}$ są bardzo duże i operowanie na nich jest bardzo kosztowne obliczeniowo. Zamiast zbioru liczb rzeczywistych używane są więc *ciała skończone*. Ciałami skończonymi są na przykład reszty z dzielenia przez p, gdzie p jest liczbą pierwszą. Ogólnie zaś jest to struktura, w której zdefiniowane są dodawanie i mnożenie spełniające oczekiwane przez nas własności, np. $a \cdot (b + c) = ab + ac$, możliwe jest zdefiniowanie dzielenia itd. Można pokazać, że dla każdej liczby

pierwszej p i dodatniej liczby naturalnej k istnieje ciało o p^k elementach. Co więcej, również wielomiany o współczynnikach z ciała skończonego mają wiele własności wspólnych z wielomianami o współczynnikach rzeczywistych, w szczególności niezerowy wielomian stopnia mniejszego niż k ma mniej niż k miejsc zerowych. I to właśnie wielomianów o współczynnikach z ciała skończonego używa się w kodach Reeda–Solomona. Zwykle używamy ciała o 2^m elementach, gdyż wtedy jeden kodowany element naturalnie odpowiada ciągowi m bitów. Zauważmy, że musimy zapewnić $n \leqslant 2^m$: nie możemy obliczyć wartości wielomianu $w(x)$ w więcej niż 2^m różnych punktach; dla ułatwienia punkty te dalej będziemy oznaczać przez $0, \ldots, n-1$.

Wiemy już, że używając kodu Reeda–Solomona, możemy poprawiać błędy, jednak na razie znamy jedynie bardzo nieefektywną procedurę poprawiania: przeglądamy kolejne możliwe wiadomości i sprawdzamy, która najmniej różni się od otrzymanej. Do przejrzenia jest $|\mathbb{F}^k|$ wiadomości, co jest bardzo kosztowne. Podamy wydajny algorytm dekodowania Welcha–Berlekampa; nie jest on najszybszy, lecz jest najprostszy w opisie i dowodzie poprawności. Niech $(u_0, \ldots, u_{n-1})$ będzie otrzymaną wiadomością, zaś $(w_0, \ldots, w_{n-1})$ słowem kodowym w odległości najwyżej $\lfloor (n-k)/2 \rfloor$; niech $w(x)$ będzie takim wielomianem stopnia mniejszego niż k, że $w(i) = w_i$ dla $i = 0, \ldots, n-1$. Zdefiniujmy też wielomian błędu $e(x)$ jako

Co ciekawe, algorytm Welcha–Berlekampa jest opatentowany: Lloyd R. Welch and Elwyn R. Berlekamp. Error correction of algebraic block codes. US Patent Number 4,633,470, December 1986.

$$e(x) = \prod_{i : u_i \neq w_i} (x - i).$$

Wtedy $\deg(e) \leqslant \lfloor (n-k)/2 \rfloor$. Wielomianu tego nie znamy, ale pozwala nam on „usunąć błędy”: zauważmy, że dla $i = 0, \ldots, n-1$ zachodzi

$$u_i e(i) = w(i) e(i) \tag{1}$$

Faktycznie: jeśli $u_i \neq w_i$, to obie strony równe są 0, w przeciwnym przypadku $u_i = w(i)$. Zdefiniujmy wielomian $q(x) = w(x)e(x)$, wtedy $\deg(q) < \lfloor (n+k)/2 \rfloor$. Teraz mamy już dwa nieznane wielomiany $q(x)$, $e(x)$, które spełniają jednak $q(x)/e(x) = w(x)$. Używając wielomianu $q(x)$, możemy uprościć (1) do

$$u_i e(i) = q(i) \text{ dla } i = 0, \ldots, n-1. \tag{2}$$

Wielomiany $q(x)$, $e(x)$ mają $\lfloor (n+k)/2 \rfloor$ oraz $\lfloor (n-k)/2 \rfloor + 1$ współczynników oraz $e(x)$ ma współczynnik wiodący 1. Potraktujmy te n nieznanych współczynników jako zmienne. Wtedy (2) można potraktować jako układ n równań na n współczynników; otrzymane równania są liniowe, tzn. każdy ze współczynników mnożymy przez ustaloną liczbę. Równania te są nad ciałem skończonym, ale większość własności układów równań nad liczbami rzeczywistymi przenosi się na układy równań nad ciałami skończonymi, w szczególności można je rozwiązać tak jak układy równań nad liczbami rzeczywistymi, np. przez podstawienia. Układ (2) ma rozwiązanie: wielomiany $w(x)$, $e(x)$, $q(x)$ są dobrze określone (choć nam nieznane), może mieć jednak wiele rozwiązań. Nie szkodzi: pokażemy, że jeśli $e'(x)$, $q'(x)$ jest innym rozwiązaniem, to $q(x)e'(x) = q'(x)e(x)$, i w takim razie $q(x)/e(x) = q'(x)/e'(x)$. Zauważmy, że z konstrukcji $\deg(q') < \lfloor (n+k)/2 \rfloor$ oraz $\deg(e') \leqslant \lfloor (n-k)/2 \rfloor$ i w takim razie $q(x)e'(x)$, $q'(x)e(x)$ mają stopień najwyżej $n-1$ i mają one takie same wartości w $i = 0, \ldots, n-1$: $q(i)e'(i) = u_i e(i) e'(i) = q'(i)e(i)$, a więc są równe. Wystarczy więc wyliczyć dowolne rozwiązanie $q'(x)$, $e'(x)$ układu (2) i wtedy $w(x) = q'(x)/e'(x)$.

Kody Reeda–Solomona znalazły szerokie zastosowanie: na płytach CD i DVD, w standardzie GSM, w transmisji z sond kosmicznych, w kodach 2D… Nie są bez wad: choć poprawiają $\lfloor (n-k)/2 \rfloor$ błędów, jednak nie na pojedynczych bitach, lecz na elementach ciała, które reprezentowane są jako wiele bitów. Ta wada staje się zaletą, gdy powstałe błędy pojawiają się „w serii”, gdyż wtedy wiele błędów wpływa na znacznie mniejszą liczbę elementów ciała. Ponadto ich gwarancje są „pesymistyczne” – poprawiają każde wystąpienie $(n-k)/2$ błędów, w wielu zastosowaniach błędy zdarzają się losowo i zwykle można poprawić ich o wiele więcej. Ale to temat na inny artykuł…