УДК 519.72

І.О. Завадський, к. ф.-м. н., доцент

Префіксний стискальний код на основі нижнього (2,3)-подання чисел

Київський національний університет імені Тараса Шевченка, 03680, м. Київ, пр-т. Глушкова 4д, e-mail: zava@ukr.net

Igor O. Zavadskyi, Associate Professor, Ph. D.

Compressing prefix code by the means of low (2,3)-representation of numbers

Taras Shevchenko National University of Kyiv, 03680, Kyiv, Glushkova av., 4d

Описано ефективний метод стискання природномовних текстів за допомогою префіксного коду, що базується на нижньому (2,3)-поданні цілих чисел. Запропонований код має приблизно таку саму ефективність, що й один із найкращих на сьогодні кодів для стискання текстів — код Фібоначчі Fib3, однак, на відміну від останнього, має властивість негайного розділення.

Ключові слова: стискальні коди,(2,3)-подання, стискання текстів, префіксний код.

The low (2,3)-representation of numbers is the alternative binary representation of numbers in the 2-radix numeration system. It has different applications in coding theory. Its properties can be used for constructing error correction or compression prefix codes. Two compression codes by the means of low (2,3)-representation of numbers are constructed in the paper. The first one, named C_1, is intended for storing the sequence of numbers and the second one (C_2) effectively compresses the natural language texts. The C_2 code is asymptotically better than the well-known Fib3 code – the best code for text compression in Fibonacci codes family and has approx. the same efficiency for realistic alphabet size. However, the C_2 code has better properties for fast search in compressed file. Namely, if some codeword has been found, it is enough to check 4 previous bits to be sure that this is true codeword entry, not the suffix of other codeword, while in Fib3 code the number of bits to be checked is not fixed in general case.

Key words: compression code, (2,3)-representation, text compression, prefix code.

Статтю представив д.ф.-м.н., проф. Анісімов А.В.

Вступ

Подання чисел у двохбазисній системі з основним базисом 2 і допоміжним базисом 3 було вперше досліджено в [1]. У [2] було введено так зване *нижнє (2,3)-подання* чисел. Опишемо стисло його сутність.

Нехай $\mathbf{N}_{2,3}$ – множина натуральних чисел, які є взаємно простими із 2 і 3, $x \in \mathrm{N}_{2,3}$, $x > 1$, $n = \lfloor \log_2 x \rfloor$. Тоді x можна подати як $2^{n-1} + 3^k x_1$ або $2^{n-2} + 3^k x_1$, де $k \in \mathrm{N}_{2,3}$, $x_1 \in \mathrm{N}_{2,3}$, $x_1 < x$ і лише в одній із цих форм. Застосовуючи аналогічний розклад до x_1, отримаємо x_2; далі обчислюватимемо x_{i+1} з рівняння $x_i = 2^{n_i} + 3^{k_i} x_{i+1}$, поки на певній ітерації t не отримаємо $x_t = 1$ або $x_t = 2$. *Нижнім (2,3)-кодом* називається будь-який двійковий код, за яким можна відновити послідовність значень $x_t, \ldots, x_1, x$.

Для однозначного декодування чисел достатньо зберігати величини $\Delta_i = \lfloor \log_2 3^{k_i} x_{i+1} \rfloor - n_i$ і k_i, які отримуємо на кожній ітерації під час розкладання x. У [2] показано, що Δ_i може набувати лише значень 0, 1 або 2.

У цій статті на основі нижнього (2,3)-подання чисел буде побудовано префіксний код, призначений для кодування послідовності двійкових чисел – назвемо його C_1. Також код C_1 буде розширено до повного префіксного коду C_2, за допомогою якого можна ефективно стискати природномовні тексти.

Код C_1

Розглянемо задачу кодування послідовності чисел. Розв'язуючи її, слід насамперед забезпечити можливість розділення чисел, тобто відділення в результуючій послідовності коду одного числа від коду іншого. З цією метою визначають *роздільник* – послідовність двійкових символів, яка може зустрічатися лише в кінці чи лише на початку кодового

© І.О. Завадський, 2014

слова. Коди з роздільником є *префіксними*, тобто такими, що жодне кодове слово не є префіксом іншого. Далі буде побудовано префіксний двійковий код на основі нижнього (2,3)-подання чисел.

Кодове слово являє собою конкатенацію (Δ,k)-блоків, тобто двійкових рядків, що складаються з двох частин – кодів величин Δ_i та k_i, отримуваних на кожній ітерації під час побудови (2,3)-розкладу числа. Блоки записують у зворотному порядку відносно того, як їх отримують під час кодування: $(\Delta_{t-1},k_{t-1}),\ldots,(\Delta_0,k_0)$. Значення Δ кодується так: $\Delta=2$ – символом 0, $\Delta=1$ – символами 11, а $\Delta=0$ – символами 10. Значення k кодується послідовністю 1…10, де кількість одиниць дорівнює k–1, за деякими винятками, що виникають внаслідок вибору роздільника.

Для утворення роздільника в кінець слова дописують символи 110. Оскільки кожен (Δ,k)-блок, а також кожне кодове слово закінчуються символом 0, то роздільником можна вважати рядок 0110. Якщо блок (Δ_0,k_0) кодового слова має вигляд 0110 або 10110, тобто $k_0=3$ і $\Delta_0\neq1$, то він уже містить роздільник і дописувати рядок 110 у кінець слова не потрібно.

Таким чином, (Δ,k)-блоки 110, 0110, 10110 є розділовими; коли якийсь із них трапляється, на ньому кодове слово і закінчується. При цьому останній блок 110 не позначає ніяких Δ й k і під час декодування має ігноруватися, а блоки 0110 та 10110 — враховуватися. Отже, жоден (Δ,k)-блок, що відповідає елементу нижнього (2,3)-розкладу, не повинен мати вигляд 110, і жоден (Δ,k)-блок, крім останнього, не повинен мати вигляд 0110 чи 10110. Однак, звичайно, отримувані в нижньому (2,3)-розкладі значення (Δ_i,k_i) можуть порушувати ці умови. А саме, ця небажана ситуація виникає, якщо:

- $\Delta=1$ і $k=1$ (тоді утворюється блок 110);
- $\Delta\neq1$, $k=3$ і (Δ,k)-блок не є останнім (це блок 0110 чи 10110).

Легко перевірити (і це показано в [2]), що для блоку (Δ_{t-1},k_{t-1}), який записується в кодовому слові першим, перший із цих двох випадків неможливий. Тому щоб уникнути зазначеної небажаної ситуації, значення k в (Δ,k)-блоці кодуватимемо послідовністю 1…10, де кількість одиниць дорівнює k, а не k–1, у таких випадках:

- $\Delta=1$ і (Δ,k)-блок не є першим;
- $\Delta\neq1$, $k\geq3$ і (Δ,k)-блок не є останнім.

У такий спосіб певний префіксний код зіставляється множині натуральних чисел, взаємно простих із 2 і 3. Числу 1, нижній (2,3)-розклад якого є пустим, зіставлено найкоротше кодове слово, що складається лише з розділової послідовності 110. Разом із останнім нулем попереднього кодового слова ця послідовність утворює роздільник. Щоб зіставити певне кодове слово довільному натуральному числу a, можна знайти число під номером a у зростаючому ряді чисел, взаємно простих із 2 й 3, та закодувати його. Таке число дорівнюватиме $3a - a \bmod 2 - 1$.

У табл. 1 наведено 15 найменших чисел, взаємно простих із 2 і 3, їхні нижні (2,3)-розклади та коди.

Табл. 1. (2,3)-розклади та коди 15 найменших чисел, взаємно простих із 2 і 3.

a	x	(Δ_0,k_0)	x_1	(Δ_1,k_1)	x_2	код
1	1					110
2	5	0, 1	1			100 110
3	7	2, 1	2			00 110
4	11	2, 2	1			010 110
5	13	1, 2	1			1110 110
6	17	0,2	1			1010 110
7	19	1, 1	5	0, 1	1	100 1110 110
8	23	0, 1	5	0, 1	1	100 100 110
9	25	2, 1	7	2, 1	2	00 00 110
10	29	1, 1	7	2, 1	2	00 1110 110
11	31	2, 3	1			0110
12	35	1, 3	1			11110 110
13	37	0, 1	7	2, 1	2	00 100 110
14	41	2, 1	11	2, 2	1	010 00 110
15	43	0, 3	1			10110

Декодування C_1

Як було зазначено вище, останнім елементом (2,3)-розкладу є число $x_t = 1$ або $x_t = 2$. З нього і починається декодування, тобто обчислення послідовності $x_t,\ldots,x_1,x_0=x$. Виконується воно так: за величинами x_{i+1}, Δ_i та k_i можна обчислити $n_i=\lfloor\log_2 3^{k_i}x_{i+1}\rfloor-\Delta_i$, а відтак – $x_i=2^{n_i}+3^{k_i}x_{i+1}$. У [2] показано, як вибирати одне з двох початкових значень: $x_t=2$ тоді й тільки тоді, коли $\Delta_{t-1}=2$ і $k_{t-1}=1$; у решті випадків $x_t=1$.

Опишемо алгоритм декодування детальніше, почавши із процедури видобування (Δ,k)-блоку, якщо відомий його перший біт. Вона очевидна: якщо перший біт дорівнює 0, він кодує значення Δ, інакше це значення кодується двома першими бітами. Після коду Δ записано код k, що являє собою послідовність одиничних

символів (можливо, порожню), доповнену розташованим за нею нульовим символом. Загальний алгоритм декодування, тобто знаходження за кодовим словом відповідного йому натурального числа a, є таким.

1. Зчитуємо перший (Δ,k)-блок. Якщо він дорівнює 110, то $x \leftarrow 1$; якщо 0110, то $x \leftarrow 31$; якщо 10110, то $x \leftarrow 43$ і в усіх цих випадках переходимо на крок 9. Інакше визначаємо значення Δ за його кодуванням і переходимо на крок 2.
2. Якщо Δ=1 або код k містить більше 2 одиниць, покладаємо значення k рівним кількості одиниць у коді k, інакше покладаємо значення k рівним кількості символів у коді k.
3. Якщо Δ=2 і k=1, покладаємо $x \leftarrow 2$, інакше – $x \leftarrow 1$.
4. $n \leftarrow \lfloor \log_2 3^k x \rfloor - \Delta$, $x \leftarrow 2^n + 3^k x$.
5. Зчитуємо черговий (Δ,k)-блок. Якщо він дорівнює 110, то переходимо на крок 9, інакше визначаємо значення Δ за його кодуванням і переходимо на крок 6.
6. Якщо зчитаний (Δ,k)-блок дорівнює 0110 або 10110, покладаємо $k \leftarrow 3$, обчислюємо $n \leftarrow \lfloor \log_2 3^k x \rfloor - \Delta$, $x \leftarrow 2^n + 3^k x$ і переходимо на крок 9. Інакше переходимо на крок 7.
7. Якщо Δ=1 або код k містить більше 2 одиниць, покладаємо значення k рівним кількості одиниць у коді k, інакше – кількості символів у коді k.
8. Переходимо на крок 4.
9. Якщо $x \bmod 3 = 1$, покладаємо $a \leftarrow (x+2)/3$, інакше $a \leftarrow (x+1)/3$. Алгоритм завершено.

(Δ,k)-код C_2

Наявність однозначного алгоритму декодування для коду C_1 означає, що цей код є роздільним. Однак він не є повним, тобто множину його кодових слів можна розширити і при цьому властивість роздільності не втратиться. Щоб показати це, побудуємо роздільний код, який містить всі кодові слова C_1, а також ще деякі. Цей код означити досить просто: він складається зі слова 110, а також усіх двійкових слів, які не починаються з послідовності 110, закінчуються послідовністю 0110 і не містять її ніде, крім кінця слова. Позначимо цей код як C_2. Очевидно, що код C_2 містить всі кодові слова коду C_1, є роздільним і роздільником у ньому, як і в коді C_1, служить послідовність 0110.

Прикладом слова, яке належить коду C_2, але не належить C_1, є 100 00 110. За наведеним вище алгоритмом воно декодується так.

1. За (Δ,k)-блоком 100 визначаємо Δ=0, k=1 і покладаємо $x \leftarrow 1$.
2. $n \leftarrow \lfloor \log_2 3^1 \cdot 1 \rfloor - 0 = 1$, $x \leftarrow 2^n + 3^k x = 5$.
3. За (Δ,k)-блоком 00 визначаємо Δ=2, k=1.
4. $n \leftarrow \lfloor \log_2 3^1 \cdot 5 \rfloor - 2 = 1$, $x \leftarrow 2^n + 3^k x = 17$.
5. (Δ,k)-блок 110 позначає кінець кодового слова та не впливає на результат декодування, що становить 17.

Однак, як видно з табл. 1, код числа x=17 зовсім інший – 1010 110 і застосування до нього алгоритму декодування також дає в результаті 17. Річ у тім, що послідовність (17, 5, 1), якій відповідає код 100 00 110, не є нижнім (2,3)-розкладом числа 17, оскільки передбачає подання числа 17 у вигляді 17 = $2^1 + 3^1 \cdot 5$, хоча $n = \lfloor \log_2 17 \rfloor = 4$ і зазначене подання має вигляд $2^{n-3} + 3^k x_1$, а не $2^{n-1} + 3^k x_1$ чи $2^{n-2} + 3^k x_1$, як мало б бути в нижньому (2,3)-розкладі. Загалом така ситуація пов'язана з тим, що, як показано в [2], у випадку, коли $7/8 \cdot 2^{m+1} < 3^k x_i < 2^{m+1}$ для деякого m, величина Δ може набувати лише двох значень: 0 і 1, але не значення 2. Тому кодові слова з C_2, які призводять до порушення цієї умови, не належать коду C_1. Так, у розглянутому вище прикладі $3^k x_1 = 15$ і вказані нерівності виконуються: 14 < 15 < 16, а отже, значення Δ_1 не може дорівнювати 2, хоча з (Δ,k)-блоку 00 ми отримуємо саме це значення.

Зазначимо, що довільне кодове слово з C_2 можна розглядати як послідовність означених вище (Δ,k)-блоків, не пов'язуючи їх, однак, із нижнім (2,3)-розкладом чисел. Нагадаємо, кодом Δ вважаємо послідовності 0, 10 та 11, а послідовність вигляду 1…10, де кількість одиниць може бути нульовою, – кодом k. Конкатенація коду Δ та коду k утворює (Δ,k)-блок.

Цілком очевидно, що будь-яку послідовність двійкових символів можна інтерпретувати, як послідовність (Δ,k)-блоків, у якій, можливо, останній блок неповний. Справді, якщо послідовність починається із символу 0, його інтерпретуємо як код значення Δ, а якщо із символу 1, то кодом Δ вважаємо перші два символи послідовності. Якщо за кодом Δ

розташовано 0, то цей символ вважаємо кодом k, а якщо 1, то кодом k вважаємо послідовність 1…10. Таким чином можна виокремити перший (Δ,k)-блок, а послідовність, що залишилася, інтерпретувати в той само спосіб знову.

У свою чергу, всі кодові слова C_2 можна інтерпретувати як послідовність (Δ,k)-блоків навіть без обтинання останнього блоку. Дійсно, є лише 3 слова з C_2, які складаються з одного (Δ,k)-блоку: 110, 0110 та 10110, а решта слів закінчуються символами 0110, перед якими записано певний префікс. Якщо цей префікс закінчується символом 1, то останні 5 символів слова – це 10110. Зауважимо, що перші 2 з цих символів (10) можуть траплятися лише або на початку (Δ,k)-блоку, або в його кінці. У першому випадку останнім блоком кодового слова є 10110, а в другому – 110 і в обох випадках перед останнім (Δ,k)-блоком записано цілу кількість інших (Δ,k)-блоків. Якщо ж префікс закінчується символом 0, то отримуємо два нулі поспіль: 00110, а така ситуація може мати місце лише якщо ці нулі являють собою повний (Δ,k)-блок або якщо перший 0 є кінцем певного (Δ,k)-блоку. У першому випадку останнім (Δ,k)-блоком кодового слова є 110, а в другому – 0110 і в обох випадках перед останнім (Δ,k)-блоком записано цілу кількість інших (Δ,k)-блоків.

Таким чином, код C_2 можна вважати *(Δ,k)-кодом*, тобто таким, що його кодові слова складаються з (Δ,k)-блоків.

Повнота коду C_2

Визначимо формулу, за якою обчислюється f_n – кількість кодових слів довжини n у коді C_2. Розглянемо деяке кодове слово x довжини n та припустимо, що його утворено додаванням зліва (Δ,k)-блоку до коротшого кодового слова y. Такий блок назвемо *нетермінальним*, оскільки він не є кінцевим блоком слова (*термінальними* вважатимемо блоки 110, 0110 та 10110, якими закінчуються кодові слова). Так, якщо $|y|=n-2$, то першим нетермінальним (Δ,k)-блоком слова x може бути тільки 00. Справедливе й зворотне твердження: з будь-якого кодового слова довжини $n-2$ можна утворити кодове слово довжини n, додавши зліва блок 00. Взагалі, якщо до кодового слова довжини $n-m$ додати зліва (Δ,k)-блок довжини m, то утвориться кодове слово довжини n. Зауважимо, що загалом кількість нетермінальних блоків така:

1 (Δ,k)-блок довжини 2: 00;

2 (Δ,k)-блоки довжини 3: 010, 100;

2 (Δ,k)-блоки довжини 4: 1110, 1010;

2 (Δ,k)-блоки довжини 5: 11110, 01110;

3 (Δ,k)-блоки довжини 6: 111110, 011110, 101110;

3 (Δ,k)-блоки довжини 7: 1111110, 0111110, 1011110 і т.д.

Загалом для $t > 6$ усі (k,Δ)-блоки довжини t утворюються з (k,Δ)-блоків довжини $t-1$ вставленням символу «1» після 2-ї позиції і тому, якщо $t \geq 6$, то кількість (k,Δ)-блоків довжини t становить 3.

Отже, якщо утворювати кодові слова з коротших слів додаванням зліва одного (Δ,k)-блоку, то, з огляду на зазначені вище кількості блоків певної довжини, отримаємо таке рекурентне співвідношення для f_n, що справедливе для $n \geq 6$:

$$f_n=f_{n-2} + 2(f_{n-3} + f_{n-4} + f_{n-5}) +3(f_{n-6}+ f_{n-7}+\ldots) \qquad (1)$$

Зведемо це співвідношення до скінченної форми:

$$f_n= f_{n-2} + 2(f_{n-3} + f_{n-4} + f_{n-5}) +3(f_{n-6}+ f_{n-7}+\ldots)= f_{n-2}+ +f_{n-3}+[f_{n-3}+2(f_{n-4}+f_{n-5}+ f_{n-6})+3(f_{n-7}+ f_{n-8}+\ldots)]+ +f_{n-6} = f_{n-1}+f_{n-2}+ f_{n-3}+ f_{n-6} \qquad (2)$$

Формула (2) справедлива для $n \geq 7$, оскільки під час її виведення за формулою (1) було виражено величину f_{n-1}, а формула (1) справедлива для $n \geq 6$.

Достатньо очевидно, що код C_2, як і C_1, є роздільним і префіксним. Однак, на відміну від коду C_1, він є також повним. Доведемо всі ці факти як теорему.

Теорема 1. Код C_2 є роздільним, префіксним і повним.

Доведення. Необхідною і достатньою умовою того, що код має зазначені в умові теореми властивості, є виконання нерівності Крафта-Макмілана як рівності: $\sum_{c\in C_2} 2^{-|c|} = 1$. Якщо f_n – кількість кодових слів довжини n, то цю рівність можна переписати у вигляді $\sum_{n=1}^{\infty} 2^{-n} f_n = 1$. Позначивши $\sum_{n=1}^{\infty} 2^{-n} f_n$ через s, врахувавши, що $f_1= f_2= 0$, $f_3= f_4=1$, $f_5=2$, $f_6=3$ та розклавши f_n за формулою (2) для $n \geq 7$, отримаємо:

$$s = \sum_{n=1}^{\infty} 2^{-n} f_n = \frac{1}{8} + \frac{1}{16} + \frac{2}{32} + \frac{3}{64} + + \sum_{n=7}^{\infty} 2^{-n} \left(f_{n-1} + f_{n-2} + f_{n-3} + f_{n-6}\right)= = \frac{19}{64} + \frac{1}{2}\sum_{n=7}^{\infty} 2^{-(n-1)} f_{n-1} + + \frac{1}{4}\sum_{n=7}^{\infty} 2^{-(n-2)} f_{n-2} +$$

$+\frac{1}{8}\sum_{n=7}^{\infty}2^{-(n-3)}f_{n-3}+$
$+\frac{1}{64}\sum_{n=7}^{\infty}2^{-(n-6)}f_{n-6}=$
$=\frac{19}{64}+\frac{1}{2}\left(s-\frac{1}{4}\right)+\frac{1}{4}\left(s-\frac{3}{16}\right)+\frac{1}{8}\left(s-\frac{1}{8}\right)+\frac{1}{64}s=$
$=\frac{7}{64}+\frac{57}{64}s.$

Звідси випливає, що $\frac{7}{64}-\frac{7}{64}s=0$, а отже s=1.

Теорему доведено.

Асимптотична щільність коду C_2

Однією з найважливіших характеристик коду з точки зору його застосування до стискання інформації є щільність, що визначається кількістю кодових слів, довжина яких не перевищує n. Позначимо цю кількість через $s_n=\sum_{i=1}^{n}f_i$. Врахуємо, що f_3= f_4=1, f_5=2, f_6=3 і, підсумовуючи за всіма індексами $n\geq7$ обидві частини формули (2), для $n\geq7$ отримаємо:

$$s_n=\sum_{i=3}^{6}f_i+\sum_{i=7}^{n}f_i=7+\sum_{i=7}^{n}(f_{i-1}+fi-2+fi-3+fi-6)\,. \quad (3)$$

Зауважимо, що
$\sum_{i=7}^{n}f_{i-1}=\sum_{i=6}^{n-1}f_i=s_{n-1}-4$
$\sum_{i=7}^{n}f_{i-2}=\sum_{i=5}^{n-2}f_i=s_{n-2}-2$
$\sum_{i=7}^{n}f_{i-3}=\sum_{i=4}^{n-3}f_i=s_{n-3}-1$
$\sum_{i=7}^{n}f_{i-6}=\sum_{i=1}^{n-6}f_i=s_{n-6}$

Підставляючи ці вирази до формули (3), отримаємо:

$$s_n=s_{n-1}+s_{n-2}+s_{n-3}+s_{n-6} \quad (4)$$

Оскільки $s_2=s_1=s_0=s_{-1}=\cdots=0$, $s_3=1$, $s_4=2$, $s_5=4$, $s_6=7$, то співвідношення (4) виконується для $n\geq6$. Знайдемо для нього твірну функцію G(z):

$$G(z)=\sum_{n=0}^{\infty}s_nz^n=z^3+2z^4+4z^5+$$
$$+n=6\infty snzn\quad=z3+2z4+4z5+n=6\infty(sn-1++sn-2+sn-3+sn-6)zn \quad (5)$$

Врахуємо, що
$\sum_{n=6}^{\infty}s_{n-1}z^n=z\sum_{n=6}^{\infty}s_{n-1}z^{n-1}=$
$=zn=5\infty snzn=zGz-z4-2z5;$
$\sum_{n=6}^{\infty}s_{n-2}z^n=z^2\sum_{n=6}^{\infty}s_{n-2}z^{n-2}=$
$=z2n=4\infty snzn=z2Gz-z5;$
$\sum_{n=6}^{\infty}s_{n-3}z^n=z^3\sum_{n=6}^{\infty}s_{n-3}z^{n-3}=$
$=z3n=3\infty snzn=z3Gz;$
$\sum_{n=6}^{\infty}s_{n-6}z^n=z^6\sum_{n=6}^{\infty}s_{n-6}z^{n-6}=$
$=z6n=0\infty snzn=z6Gz.$

Підставляючи ці співвідношення у формулу (5) та розв'язуючи отримане рівняння відносно $G(z)$, матимемо:

$$G(z)=\frac{z^3+z^4+z^5}{1-z-z^2-z^3-z^6}=\frac{z^3}{1-2z+z^3-z^4}.$$

Розкладемо у суму найпростіших дробів:

$$G(z)=\frac{-0{,}3618+0{,}2982i}{z-0{,}809-0{,}9816i}+\frac{-0{,}3618-0{,}2982i}{z-0{,}809+0{,}9816i}-$$
$$-\frac{0{,}1888}{z+1{,}1537}-\frac{0{,}0876}{z-0{,}5357} \quad (6)$$

Як видно з (5), коефіцієнт s_n дорівнює n-му коефіцієнту ряду Маклорена для функції $G(z)$. Якщо розкладати в ряд Маклорена функцію $g(z)=\frac{1}{z-a}$, то n-й член ряду дорівнюватиме $\frac{x^n}{n!}g^{(n)}(0)=\frac{(-1)^nn!x^n}{n!(-a)^n}=\frac{x^n}{a^n}$. Таким чином, порядок росту величини s_n визначатиметься як $\frac{1}{a^n}$, де величину a слід вибирати за умовою, що $|a|$ є найменшим серед усіх дробів вигляду $\frac{b}{z-a}$ у формулі (6). Це останній дріб, тому a=0,5357 і порядок росту величини s_n становить

$$\left(\frac{1}{0{,}5357}\right)^n\approx1{,}867^n \quad (7)$$

Застосування коду C_2 для стискання природномовних текстів

Для визначення ефективності стискання природномовних текстів, як правило, припускають, що слова в мові розподілено за законом Зіпфа: якщо обсяг словника становить N і його впорядковано за спаданням частот слів, то ймовірність того, що певне слово в тексті буде займати i-у позицію у словнику, становить $p_i=1/iH_N$, де $H_N=\sum_{j=1}^{N}1/j\approx\ln N++0{,}5772$ – N-е гармонійне число. Ефективність кодування текстів мовою, словник якої містить N слів, визначається середньою довжиною кодового слова, що дорівнює

$$\sum_{i=1}^{N}p_i|c_i|=\frac{1}{H_N}\sum_{i=1}^{N}\frac{1}{i}|c_i| \quad (8)$$

Нехай s_n – кількість кодових слів, довжина яких не перевищує n, а M_N дорівнює найменшому значенню n, за якого $s_n\geq N$. Покладемо також

$$q_n=\begin{cases}s_n,\text{якщо } s_n\leq N\\ N,\text{якщо } s_n>N\end{cases}$$

Тоді формулу (8) можна переписати у вигляді

$$\frac{1}{H_N}\sum_{k=1}^{M_N}k\sum_{i=q_{k-1}+1}^{q_k}\frac{1}{i}\approx\frac{1}{H_N}\sum_{k=1}^{M_N}k\ln\frac{q_k}{q_{k-1}+1} \quad (9)$$

Розраховані за формулою (9) середні довжини кодових слів для різних кодів наведено в табл. 2. Серед кількох класів кодів, що застосовуються для стискання текстової інформації, до розглянутих у цій статті кодів найближчими є коди Фібоначчі, детально

досліджені в [3], оскільки вони також є префіксними кодами із підтримкою синхронізації, у яких довжина кодового слова не дорівнює цілому числу байтів. Як показано у [3], серед цього сімейства кодів найбільший коефіцієнт стискання у разі застосування до природномовних текстів має код Fib3. Асимптотична щільність цього коду становить $(1{,}839)^n$, тобто є меншою за асимптотичну щільність коду C_2 (див. (7)). Як видно з табл. 2, у разі розподілу частот елементів алфавіту за законом Зіпфа код C_2 матиме вищу щільність, ніж Fib3, для обсягу алфавіту від 100 млн. слів і вище. Те, що ступінь стискання великих алфавітів кодом C_2 буде кращим за аналогічний показник коду Fib3, видно також із того простого факту, що кількість слів довжини n у коді C_2 визначається за формулою (2): $f_n=f_{n-1}+f_{n-2}+f_{n-3}+f_{n-6}$, в той час як для коду Fib3 вона становить $f_{n-1}+f_{n-2}+f_{n-3}$. Для більш реалістичних для природних мов обсягів словників (від 100 тис. до 10 млн. слів) код C_2 має дещо нижчу щільність, ніж Fib3, однак погіршення є незначним і не перевищує 1 %.

Однак у коду Fib3, порівняно з кодом C_2, є принаймні один недолік, який стосується властивості негайного розділення, важливої для пошуку слів у стиснутому файлі без його розархівування. Як Fib3, так і C_2, а також всі інші застосовувані для стискання текстів коди характеризуються тим, що якщо в стиснутому файлі знайдено входження певного слова w, то ми не можемо гарантувати, що це входження справді відповідає слову w, оскільки воно може бути суфіксом іншого слова. У коді C_2 для перевірки, чи є w справді окремим словом, достатньо переглянути 4 біти, які передують w: якщо це біти 0110, то w – окреме слово, інакше – ні. Проте в коді Fib3 перевірки будь-якої фіксованої кількості бітів, що передують кодовому слову, недостатньо, оскільки роздільником і найкоротшим словом у цьому коді є 111, а кілька таких слів можуть «склеюватися», якщо записані одне за одним. Як один зі способів уникнення цієї проблеми, у [3] пропонується вилучати з коду Fib3 найкоротше кодове слово 111 – такий код позначимо як Fib3$^-$. Однак щільність коду Fib3$^-$ є суттєво гіршою за щільність коду C_2 навіть за реалістичних обсягів алфавітів, елементи яких розподілено за законом Зіпфа (табл. 2, за базу порівняння відносних величин взято код Fib3).

Табл. 2. Середня довжина кодового слова

Обсяг алфавіту	Fib3	Fib3$^-$	C_2
1000	8,21	8,88 (+8%)	8,35 (+1,7%)
10 000	10,1	10,61 (+5%)	10,23 (+1,3%)
100 000	11,99	12,4 (+3,4%)	12,09 (+0,8%)
1 млн	13,87	14,22 (+2,5%)	13,94 (+0,5%)
10 млн	15,76	16,06 (+1,9%)	15,8 (+0,2%)
100 млн	17,65	17,91 (+1,5%)	17,65 (0 %)

Висновки

Властивості (2,3)-подання чисел дають змогу сконструювати стискальний код, який усуває деякі недоліки відомого коду Фібоначчі Fib3, майже не знижуючи його стискальної ефективності у разі застосування до природномовних текстів. За розглянутими в статті принципами може бути сконструйовано також інші стискальні коди, що можуть виявитися ще ефективнішими для алфавітів певного обсягу.

Список літератури

1. *Anisimov A.V.* Prefix Encoding by Means of the 2,3-Representation of Numbers / A.V. Anisimov // IEEE Transactions on Information Theory. – 2013. – No 4. – P. 2359-2374.

2. *Анисимов А.В.* Помехоустойчивое префиксное кодирование на основе нижнего (2,3)-представления чисел / А.В. Анисимов, И.А. Завадский // Кибернетика и системный анализ. – 2014. – № 2. – С. 3–14.

3. *Klein S.T.* On the usefulness of Fibonacci compression codes / S.T. Klein, M.K. Ben-Nissan // Computer Journal. – 2010. – vol. 53, no. 6, p. 701–716.

References

1. ANISIMOV, A.V. (2013) Prefix Encoding by Means of the 2,3-Representation of Numbers, *IEEE Transactions on Information Theory*. vol. 59, no 4, pp. 2359-2374.

2. ANISIMOV, A.V., ZAVADSKYI, I.O. (2014) Pomekhoustojchivoe prefiksnoe kodirovanie na osnove nizhnego (2,3)-predstavlenija chisel. *Kibernetika i systemnyj analiz*. No 2, pp. 3–14.

3. KLEIN, S.T., BEN-NISSAN, M.K. (2010) On the usefulness of Fibonacci compression codes. *Computer Journal*. vol. 53, no. 6, pp. 701–716.

Надійшла до редколегії 17.12.2014