УДК 519.171.2 DOI: 10.18698/0236-3933-2017-3-33-46

АЛГОРИТМ КОДИРОВАНИЯ БИНАРНОГО ДЕРЕВА С МИНИМАЛЬНОЙ ИЗБЫТОЧНОСТЬЮ

В.П. Корвяков vladimir.korviakov@gmail.com

ПАО «Ракетно-космическая корпорация «Энергия» им. С.П. Королёва», Королёв, Московская обл., Российская Федерация

Аннотация

Предложен алгоритм кодирования и декодирования бинарных деревьев с минимальной избыточностью, т. е. использующий битовые последовательности минимально возможной длины. Рекурсивное вычисление индексов деревьев выполнено с использованием чисел Каталана. Приведены оценки сложности алгоритма. Разработанный метод позволяет построить бинарное дерево на основе уникального индекса его структуры и числа узлов без необходимости явного перечисления всех возможных структур деревьев. Минимальная длина битовой последовательности дает возможность оптимизировать деревья с заданным числом узлов с помощью генетических алгоритмов

Ключевые слова

Бинарное дерево, числа Каталана, алгоритм кодирования, минимальная избыточность

Поступила в редакцию 30.09.2016

Введение. В процессе работы над методами оптимизации и автоматической генерации графических пользовательских интерфейсов автор настоящей статьи столкнулся с проблемой кодирования структуры данных, представляющей компоновку элементов графического интерфейса с помощью бинарного дерева. В принятой модели интерфейс представлен набором виджетов (элементов интерфейса), положение и размеры которых определены рекурсивным раздвоением прямоугольных областей в вертикальном или горизонтальном направлении. При такой компоновке виджеты ассоциируются с листовыми узлами строго бинарного дерева.

Задача осложняется тем, что рассматриваемая структура данных должна быть закодирована в битовую последовательность фиксированной (для заданного числа виджетов) длины так, чтобы изменение любого ее бита преобразовывало последовательность в непротиворечивую битовую последовательность, также кодирующую интерфейс с таким же числом виджетов. Не должно возникать противоречий в битовой последовательности при скрещивании двух различных битовых последовательностей (рис. 1). Это ограничение связано с тем, что битовая последовательность должна быть использована в качестве хромосомы генетического алгоритма, который оптимизирует интерфейс с позиции заданного критерия [1]. Для ускорения работы генетического алгоритма длина хромосомы должна быть минимальной, а ее содержание — непротиворечивым.

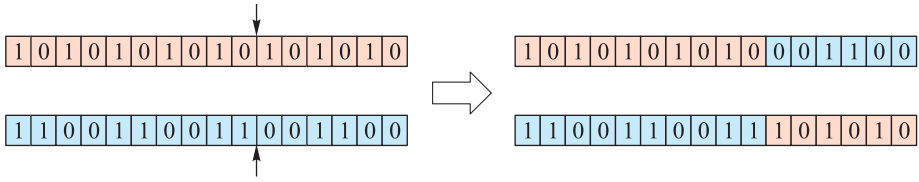

Рис. 1. Формирование хромосом при скрещивании в процессе работы генетического алгоритма

Требование существования строго двух поддеревьев у каждого нелистового узла является ограничением, которое можно снять, если рассмотреть строго бинарное дерево с $N+1$ листьями как расширенное к обычному бинарному дереву с N узлами. При этом N узлов обычного бинарного дерева можно ассоциировать с «внутренними» (т. е. нелистовыми) узлами исходного дерева (рис. 2). В работе [2] Д. Кнут показывает, что такая трансформация бинарных деревьев является однозначной и взаимной. С учетом изложенного, задача кодирования строго бинарного дерева с N+1 листьями сводится к кодированию обычного бинарного дерева с N узлами. Далее перейдем к задаче кодирования обычных бинарных деревьев с N узлами и будем иметь в виду ее эквивалентность исходной задаче кодирования строго бинарных деревьев с N+1 листьями.

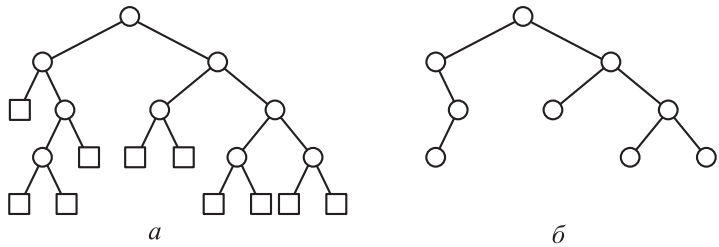

Рис. 2. Схемы преобразований расширенного (*а*) и обычного (*б*) бинарных деревьев

Метод решения задачи. Задача разработки алгоритма минимального и непротиворечивого кодирования бинарных деревьев представляет отдельный интерес. Последовательность битов должна кодировать любую структуру бинарного дерева с заданным числом узлов N. Число неизоморфных бинарных деревьев с N узлами равно N-му числу Каталана [3]:

$$C_N = \frac{1}{N+1}\binom{2N}{N} = \frac{(2N)!}{(N+1)!N!}. \qquad (1)$$

Если рассмотреть бинарное дерево, каждая структура которого имеет равную вероятность, как сообщение, то его энтропия равна

$$H = -\sum_{i=1}^{C_N} p_i \log_2 p_i = -\sum_{i=1}^{C_N} \frac{1}{C_N} \log_2 \left(\frac{1}{C_N}\right) = \log_2 C_N.$$

Откуда следует, что число битов, необходимое для кодирования структуры бинарного дерева с N узлами, равно $\lceil \log_2 C_N \rceil$, где скобки обозначают округление до ближайшего целого в большую сторону. Например, хранение бинарного дерева с девятью узлами потребует $\lceil \log_2 C_9 \rceil = \lceil \log_2 4862 \rceil = 13$ бит. Последовательность длиной $\lceil \log_2 C_N \rceil$ вследствие округления в большую сторону обладает некоторой избыточностью информации по отношению к исходному сообщению. Величина абсолютной избыточности равна

$$R_a = \lceil \log_2 C_N \rceil - \log_2 C_N.$$

Относительная избыточность составляет

$$R = \frac{\lceil \log_2 C_N \rceil - \log_2 C_N}{\lceil \log_2 C_N \rceil}.$$

На практике это будет выражаться в том, что последние $2^{\lceil \log_2 C_N \rceil} - C_N$ сообщений не будут соответствовать никакой структуре. Для того чтобы получить из исходного сообщения $I \in \left[0;\ 2^{\lceil \log_2 C_N \rceil} - 1\right]$ валидный индекс бинарного дерева, необходимо вычислить его как остаток от деления I на C_N. Здесь и далее битовая последовательность длиной L будет отождествляться с соответствующим ей целым числом $I \in \left[0;\ 2^L - 1\right]$. Кроме того, будем использовать индексацию, начинающуюся с нуля.

Следует отметить, что существуют различные методы кодирования бинарных деревьев. Этой задаче посвящена, например, работа [4]. Предложенный в этой работе метод требует строго $2N$ бит данных для кодирования бинарных деревьев с N узлами. Это основано на аппроксимации чисел Каталана [5]:

$$C_N \approx \frac{4^N}{N\sqrt{\pi N}};$$

$$\log_2 C_N \approx \log_2 4^N - \frac{3}{2}\log_2 N - \frac{1}{2}\log_2 \pi < 2N.$$

Другой метод кодирования деревьев, позволяющий кодировать деревья с N узлами последовательностью $2N + o(N)$ бит, представлен в работе [6]. Достоинство этого метода состоит в том, что он позволяет выполнять операции с закодированным деревом за время $O(1)$. Еще более компактное представление, требующее $2N + N/(\log_2 N)^{O(1)}$ бит данных, описано в работе [7]. Достаточно полный обзор методов кодирования приведен в работе [8]. Основной вывод указанной работы заключается в том, что универсального способа кодирования деревьев не существует, и для конкретных задач необходимо выбирать наиболее подходящие методы кодирования. Тем не менее до сих пор вопрос кодирования с минимально теоретически возможной избыточностью не был достаточно хорошо проработан.

Поскольку число возможных структур бинарных деревьев с N узлами равно числу Каталана C_N, любое число от 0 до $C_N - 1$ можно сопоставить с конкретной структурой дерева. Задача кодирования и декодирования заключается в нахождении алгоритма однозначного взаимного преобразования индекса дерева в структуру бинарного дерева.

Для решения этой задачи примем, что индексу 0 соответствует бинарное дерево с N узлами, полностью ориентированное по левую сторону, а индексу C_{N-1} — по правую. Установим, что каждый узел определяется следующей структурой:

```
NODE {
        NODE LLINK;
        NODE RLINK;
}
```

где *LLINK* — корневой узел левого поддерева или *EMPTY*, если левое поддерево отсутствует; *RLINK* — корневой узел правого поддерева или *EMPTY*, если правое поддерево отсутствует.

Природа древовидных структур подсказывает возможность применения рекурсивного алгоритма. На каждом вызове функции *NODE* = *decode* (I, N) по заданному индексу структуры I и числу узлов N необходимо следующее.

1. Если $N = 0$, то вернуть *EMPTY*, иначе перейти к п. 2.

2. Инициализировать структуру возвращаемого узла *NODE ROOT* {*LLINK* = = *EMPTY*, *RLINK* = *EMPTY*}.

3. Если N равно 1, то вернуть *ROOT*, иначе перейти к п. 3.

4. Вычислить число узлов левого поддерева N_L.

5. Определить число узлов правого поддерева N_R, которое всегда равно $N - N_L - 1$.

6. Рассчитать индекс структуры левого поддерева I_L такой, что $0 \le I_L \le C_{N_L} - 1$.

7. Вычислить индекс структуры правого поддерева I_R такой, что $0 \le I_R \le C_{N_R} - 1$.

8. Вызвать функцию *decode* (I_L, N_L) и присвоить результат ее выполнения левому поддереву данного узла: *ROOT.LLINK* = *decode* (I_L, N_L).

9. Вызвать функцию *decode* (I_R, N_R) и присвоить результат ее выполнения левому поддереву данного узла: *ROOT.RLINK* = *decode* (I_R, N_R).

10. Вернуть структуру *ROOT*.

Отдельную задачу представляет вычисление числа N_L. К ее решению можно подойти исходя из следующих соображений. Поскольку нулевому индексу соответствует дерево, полностью ориентированное по левую сторону, для $I = 0$ имеем $N_L = N - 1$, $N_R = 0$. Число комбинаций левого поддерева равно C_{N-1}. Таким образом, первые C_{N-1} индексов перечисляют различные комбинации левого поддерева, состоящего из $N - 1$ узлов, при пустом правом поддереве (рис. 3, *а*).

При этом левое поддерево комбинации $C_{N-1}-1$ будет полностью ориентировано по правую сторону. На индексе C_{N-1} один узел должен быть перенесен из левого в правое поддерево: $I = C_{N-1}$; $N_L = N-2$; $N_R = 1$. Следующие C_{N-2} индексы перечисляют различные комбинации левого поддерева ($N_L = N-2$), при правом поддереве, состоящем из одного узла (рис. 3, *б*, *в*).

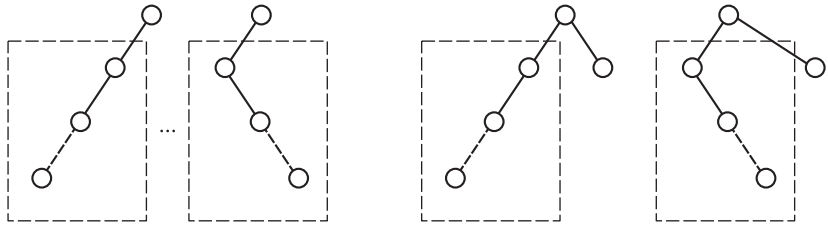

Рис. 3. Первые C_{N-1} комбинаций дерева (*а*) и комбинации с C_{N-1} (*б*) по $C_{N-1} + C_{N-2} - 1$ (*в*)

На комбинации $C_{N-1}+C_{N-2}$ еще один узел должен быть перенесен из левого в правое поддерево: $I = C_{N-1}+C_{N-2}$; $N_L = N-3$, $N_R = 2$. Начиная с $N_R = 2$ число возможных структур правого поддерева отлично от единицы и равно C_{N_R}. В общем случае это правило можно распространить и на $N_R = \{0,1\}$, так как $C_0 = C_1 = 1$. Таким образом, для каждого N_L от $N-1$ до 0 число различных комбинаций левого и правого поддеревьев равно $C_{N_L}C_{N-N_L-1}$. Это соответствует формуле рекуррентного соотношения для вычисления чисел Каталана: $C_0 = 1$ и $C_N = \sum_{i=0}^{N-1} C_i C_{N-i-1}$ для $N > 0$. С учетом изложенного, для получения чисел N_L и N_R необходимо найти такое *K*, что

$$\sum_{i=0}^{K} C_{N-i-1}C_i < I; \qquad (2)$$

$$\sum_{i=0}^{K+1} C_{N-i-1}C_i > I;$$

$$N_R = K;$$

$$N_L = N-K-1.$$

Для определения параметров I_L и I_R необходимо предварительно вычислить индекс текущей комбинации для заданных значений N_L и N_R: $\hat{I} = I - \sum_{i=0}^{K} C_{N-i-1}C_i$. Значение $\hat{I}$ равно разности заданного индекса комбинации и индекса последней комбинации, на которой было выполнено перемещение узла из левого поддерева в правое (т. е. уменьшение числа N_L и увеличение числа N_R).

При перечислении комбинаций дерева (с заданными N_L и N_R) правое поддерево фиксируется, и последовательно рассматриваются C_{N_L} комбинаций левого поддерева. Далее осуществляется переход к следующей комбинации правого поддерева и процедура повторяется. Следовательно,

$$I_R = \left\lfloor \frac{\hat{I}}{C_{N_L}} \right\rfloor;$$

$$I_L = \hat{I} - I_R C_{N_L}.$$

Скобки $\lfloor\ \rfloor$ обозначают неполное частное, т. е. округление до ближайшего целого в меньшую сторону. Таким образом, алгоритм функции *decode* можно описать следующим псевдокодом:

```
NODE decode(I, N) {
    if (N == 0) return EMPTY; // Дерево с 0 узлами – пустое
    NODE ROOT { LLINK = EMPTY; RLINK = EMPTY };
    if (N == 1) return ROOT; // Один узел без поддеревьев
    NL = N – 1;
    NR = 0;
    SUM = 0;
    OLDSUM = 0;
    K = 0;
    CL = 1;
    CR = 1;
    while (SUM <= I) {
        NL = N – K – 1;
        NR = K;
        CL = catalan(NL);
        CR = catalan(NR);
        OLDSUM = SUM;
        SUM = SUM + CL*CR;
        K++;
    }
    I = I – OLDSUM;
    IR = floor(I/CL);
    IL =      I – IR*CL;
    ROOT.LLINK = decode(IL, NL);
    ROOT.RLINK = decode(IR, NR);
    return ROOT;
}
```

В приведенном псевдокоде функция *floor(X)* выполняет округление величины *X* до ближайшего целого в меньшую сторону, а функция *catalan(N)* — вычисление *N*-го числа Каталана.

Решив задачу декодирования индекса в структуру дерева, перейдем к обратной задаче: по заданной структуре бинарного дерева с N узлами необходимо получить число I от 0 до $C_N - 1$, кодирующее такую структуру. Хотя кодирование деревьев и не требовалось для исходной задачи оптимизации графических интерфейсов, его следует рассмотреть для полноты проработки вопроса и для возможности применения этого метода при решении других проблем.

Рассмотрим бинарное дерево с N узлами, в левом поддереве которого находится N_L узлов, а в правом — $N_R = N - N_L - 1$. Принятый метод перечисления деревьев (от полностью левоориентированного к полностью правоориентированному с перебором комбинаций левого поддерева при фиксированном правом) предполагает, что начальный индекс дерева с N_R узлами левого поддерева равен $I_0 = \sum_{k=0}^{N_R-1} C_{N-k-1}C_k$. Дерево, имеющее индекс I_0, показано на рис. 4, *а*.

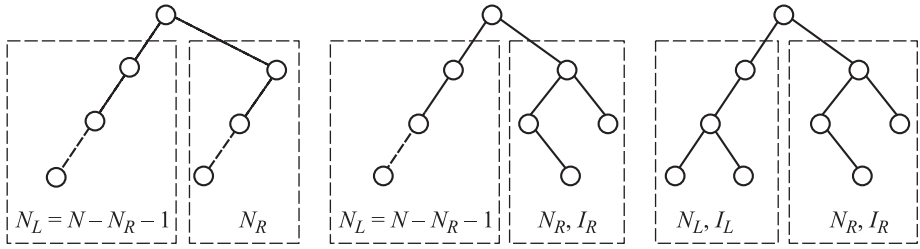

Рис. 4. Дерево с индексом $I_0 = \sum_{k=0}^{N_R-1} C_{N-k-1}C_k$ (*а*), с индексом $I = \left(\sum_{k=0}^{N_R-1} C_{N-k-1}C_k\right) + C_{N_L}I_R$ (*б*), правая часть которого приняла требуемый вид, требуемое дерево (*в*) с индексом I_R правого поддерева и I_L левого поддерева

После этого для каждой из I_R зафиксированных комбинаций правого поддерева последовательно перечисляются по C_{N_L} комбинаций левого поддерева (рис. 4, *б*). Таким образом, правое поддерево принимает требуемый вид, а левое поддерево полностью ориентировано по левую сторону. Индекс этого дерева равен $I = \left(\sum_{k=0}^{N_R-1} C_{N-k-1}C_k\right) + C_{N_L}I_R$.

Для получения исходного дерева необходимо перечислить еще I_L комбинаций левого поддерева (рис. 4, *в*). Таким образом, окончательно имеем формулу кодирования дерева:

$$I = \left(\sum_{k=0}^{N_R-1} C_{N-k-1}C_k\right) + C_{N_L}I_R + I_L.$$

Значения индексов I_R и I_L рекурсивно определяются этим же алгоритмом, либо равны нулю для пустых поддеревьев.

Алгоритм функции *encode* описан следующим псевдокодом:

```
INTEGER encode(ROOT) {
    if (ROOT == EMPTY) return 0 // Пустое дерево кодируется нулём
    N = size(ROOT) // Число узлов дерева ROOT
    NL = size(ROOT.LLINK) // Число узлов левого поддерева
    NR = N – NL – 1 // Число узлов правого поддерева
    I = 0
    for (k = 0; k < NR; k++) {
        I = I + catalan(N – k – 1)*catalan(k)
    }
        IL = encode(ROOT.LLINK) // Код левого поддерева
    IR = encode(ROOT.LLINK) // Код правого поддерева
    I = I + catalan(NL)*IR + IL
    return I
}
```

Вычислительная сложность алгоритма. Для анализа вычислительной сложности разработанных алгоритмов сделаем допущение о том, что вычисление факториалов и чисел Каталана выполняется «наивным» способом, приведенным ниже.

1. Функция факториала требует N–1 операций умножения, т. е. имеет линейную сложность $O(N)$.

2. В соответствии с формулой (1) вычисление числа Каталана требует $2N-1+N+N-1+2=4N$ операций умножения и деления, т. е. также имеет линейную сложность $O(N)$.

Кроме того, значения чисел Каталана рассчитываются каждый раз и не кэшируются. Поскольку индекс дерева может принимать любое значение от 0 до C_N-1, на каждом этапе рекурсивного вызова функции *decode* заранее неизвестно, какие индексы будут переданы функциям, вызываемым для правого и левого поддеревьев. Также неизвестно, на какой итерации (от 0 до N–1) остановится цикл поиска суммы, вычисляемой по формуле (2). Для упрощения будем полагать, что на каждом этапе дерево разбивается на две равные части, а вычисление формулы (2) в худшем случае проходит все N итераций. В соответствии с основной теоремой о рекуррентных соотношениях (*Master theorem*) [9], время работы рекуррентного алгоритма может быть оценено по формуле

$$T(n)=aT\left(\frac{n}{b}\right)+f(n),$$

где n — размер задачи; a — число подзадач в рекурсии (в рассматриваемом случае a = 2); n/b — размер каждой подзадачи (с учетом принятых допущений b = 2); $f(n)$ — сложность алгоритма, выполняемого вне рекурсивных вызовов.

В случае функции *decode* основные вычисления вне рекурсивных вызовов выполняются при расчете формулы (2). На каждом этапе суммирования вычисления произведения $C_i C_{N-i-1}$, что имеет суммарную сложность $O(N)$. С учетом допущения о том, что поиск останавливается после N итераций, функция $f(n)$ пропорциональна n^2. Для заданных условий выполняются следующие условия:

$$T(n) = 2T\left(\frac{n}{2}\right) + n^2;$$

$$f(n) = \Omega(n^c), \ c = 2;$$

$$c > \log_b a, \ a = b = 2;$$

$$af\left(\frac{n}{b}\right) \le kf(n) \text{ для } k = 0,5.$$

Эти условия соответствуют третьей форме основной теоремы о рекуррентных соотношениях, при которой время работы функции *decode* имеет асимптотическую оценку $T(n) = \Theta(f(n)) = \Theta(n^2)$.

Для функции *encode* при тех же допущениях можно применить рассуждения, аналогичные рассуждениям, использованным для функции *decode*, и оценка времени работы алгоритма также равна $\Theta(n^2)$.

Результаты. Примеры использования разработанного метода для перечисления всех структур бинарных деревьев с тремя и четырьмя узлами приведены на рис. 5 и рис. 6. Это перечисление было проведено программно, пример реализации алгоритма на языке *Python* можно найти в работе [10]. Для визуализации деревьев были использованы язык разметки *dot* [11] и система *Graphviz*. Каждое показанное на рисунках бинарное дерево было построено на основании индекса I и числа узлов N.

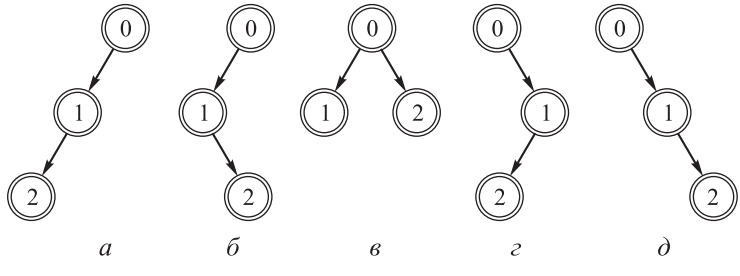

Рис. 5. Бинарные деревья, построенные на основании индекса I и числа узлов $N = 3$ (в скобках указаны представления индексов в двоичной системе счисления):
а — $I = 0$ (000 *b*); *б* — $I = 1$ (001 *b*); *в* — $I = 2$ (010 *b*); *г* — $I = 3$ (011 *b*); *д* — $I = 4$ (100 *b*)

Сравнение размеров битовых последовательностей при кодировании алгоритмом, требующим ровно $2N$ бит (L_1), и кодировании разработанным алгоритмом (L_2), а также относительные избыточности кода для существующего (R_1) и

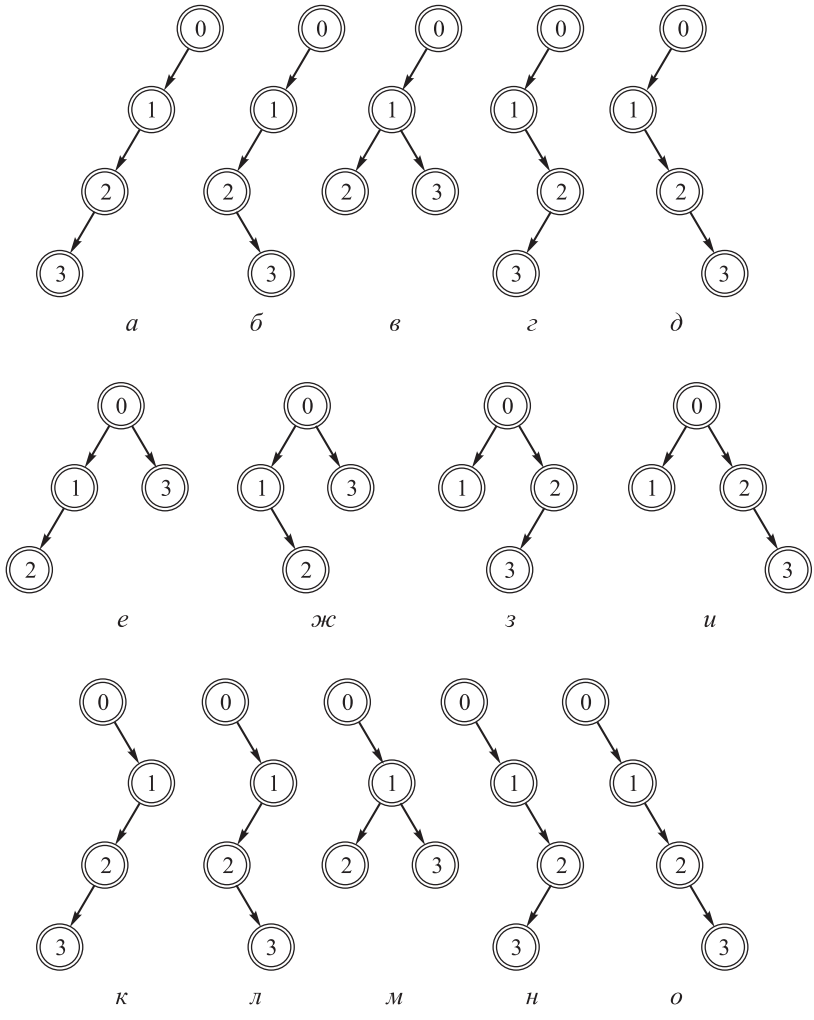

Рис. 6. Бинарные деревья, построенные на основании индекса I и числа узлов $N = 4$ (в скобках указаны представления индексов в двоичной системе счисления):

а — $I = 0$ (0000 *b*); *б* — $I = 1$ (0001 *b*); *в* — $I = 2$ (0010 *b*); *г* — $I = 3$ (0011 *b*); *д* — $I = 4$ (0100 *b*); *е* — $I = 5$ (0101 *b*); *ж* — $I = 6$ (0110 *b*); *з* — $I = 7$ (0111 *b*); *и* — $I = 8$ (1000 *b*); *к* — $I = 9$ (10001 *b*); *л* — $I = 10$ (1010 *b*); *м* — $I = 11$ (1011 *b*); *н* — $I = 12$ (1100 *b*); *о* — $I = 13$ (1101 *b*)

разработанного (R_2) алгоритмов приведены в таблице. Сравнение показывает существенный выигрыш в размере кодовой последовательности разработанного алгоритма по сравнению с существующим методом, особенно для деревьев небольших размеров.

Сравнение размеров битовых последовательностей существующего и разработанного алгоритмов

N	$L_1 = 2N$	$L_2 = \lceil \log_2 C_N \rceil$	R_1	R_2
2	4	1	0,75	0
3	6	3	0,613	0,226

Окончание таблицы

N	$L_1 = 2N$	$L_2 = \lceil \log_2 C_N \rceil$	R_1	R_2
4	8	4	0,524	0,048
5	10	6	0,461	0,101
6	12	8	0,413	0,119
7	14	9	0,375	0,028
8	16	11	0,345	0,047
9	18	13	0,32	0,058
10	20	15	0,298	0,064
20	40	33	0,185	0,012
30	60	52	0,137	0,005
40	80	72	0,111	0,012
50	100	91	0,093	0,004
60	120	111	0,081	0,006
70	140	130	0,072	0,0003
80	160	150	0,065	0,002
90	180	170	0,059	0,003
100	200	190	0,054	0,004
200	400	388	0,031	0,0008
300	600	587	0,022	0,0003
400	800	787	0,017	0,001
500	1000	986	0,014	0,0003
600	1200	1186	0,012	0,0006
700	1400	1385	0,011	0,000004
800	1600	1585	0,01	0,0002
900	1800	1785	0,009	0,0003
1000	2000	1985	0,008	0,0004

Заключение. Разработан алгоритм, позволяющий кодировать структуры бинарных деревьев с помощью битовых последовательностей минимальной длины. Длина битовой последовательности разработанного кода близка к минимальному теоретически возможному значению: $2n - \Theta(\log_2 n)$ бит. Недостатком разработанного метода является невозможность эффективного выполнения операций с закодированным представлением дерева без его предварительного декодирования и преобразования в более удобное представление. Тем не менее это кодирование было применено автором в хромосоме генетического алгоритма, оптимизирующего графический пользовательский интерфейс. Кроме того, алгоритм может быть использован в системах связи или сжатия информации. Известные методы однозначного преобразования бинарных деревьев в обычные деревья и другие смежные объекты [2, 12] позволяют кодировать любые древовидные структуры и леса с использованием разработанного алгоритма.

ЛИТЕРАТУРА

1. *Корвяков В.П.* Метод нейро-нечеткой оценки пригодности использования графического интерфейса пользователя // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2016. № 5. С. 61–74. DOI: 10.18698/0236-3933-2016-5-61-74

2. *Кнут Д.* Искусство программирования. Т. 1. Основные алгоритмы / пер. с англ. М.: Вильямс, 2006. 720 с.

3. *Davis T.* Catalan numbers // geometer.org: веб-сайт. URL: http://www.geometer.org/mathcircles/catalan.pdf (дата обращения: 17.08.2016).

4. *Bege A., Kasa Z.* Coding objects related to Catalan numbers // Studia Universitatis Babes-Bolyai. Informatica. 2001. Vol. 46. No. 1. P. 31–40. URL: http://www.cs.ubbcluj.ro/~studia-i/2001-1/3-Kasa.pdf

5. *Flajolet P., Sedgewick R.* Analytic combinatorics. Cambridge University Press, 2009. 826 p.

6. *Farzan A., Munro J.I.* A uniform paradigm to succinctly encode various families of trees // Algorithmica. 2014. Vol. 68. No. 1. P. 16–40. DOI: 10.1007/s00453-012-9664-0

URL: http://link.springer.com/article/10.1007%2Fs00453-012-9664-0

7. *Davoodi P., Raman R., Satti S.R.* On succinct representations of binary trees // arXiv.org: Cornell University Library. URL: http://arxiv.org/abs/1410.4963 (дата обращения: 29.09.2016).

8. *Mäkinen E.* A survey on binary tree coding // The Computer Journal. 1991. Vol. 34. No. 5. P. 438–443. DOI: 10.1093/comjnl/34.5.438 URL: https://academic.oup.com/comjnl/article-abstract/34/5/438/553944/A-Survey-on-Binary-Tree-Codings?redirectedFrom=fulltext

9. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ / пер. с англ. М.: Вильямс, 2013. 1328 с.

10. *Catalantree* — Binary trees decoding/encoding with Catalan numbers based algorithm // GitHub: веб-сайт. URL: https://github.com/HapKoM/catalantree (дата обращения: 24.08.2016).

11. *Gansner E., Koutsofios E., North E.* Drawing graphs with dot // Graphviz: веб-сайт: http://www.graphviz.org/Documentation/dotguide.pdf (дата обращения: 24.08.2016).

12. *Кнут Д.* Искусство программирования. Т. 4. Вып. 4. Генерация всех деревьев. История комбинаторной генерации / пер. с англ. М.: Вильямс, 2007. 160 с.

Корвяков Владимир Петрович — аспирант, инженер-программист научно-технического центра ПАО «РКК «Энергия» им. С.П. Королёва» (Российская Федерация, 141070, Московская обл., Королёв, ул. Ленина, д. 4а).

Просьба ссылаться на эту статью следующим образом:

Корвяков В.П. Алгоритм кодирования бинарного дерева с минимальной избыточностью // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2017. № 3. С. 33–46. DOI: 10.18698/0236-3933-2017-3-33-46

AN ALGORITHM OF BINARY TREE ENCODING WITH MINIMUM REDUNDANCY

V.P. Korvyakov vladimir.korviakov@gmail.com

S.P. Korolev Rocket and Space Corporation Energia, Korolev, Moscow Region, Russian Federation

Abstract

This paper proposes an algorithm of binary trees encoding and decoding with minimum redundancy, i. e. with bit sequences of minimal possible length. We performed recursive computation of trees indices using Catalan numbers. Moreover, we give the algorithm complexity estimations. The developed method allows us to construct a binary tree on the basis of unique index of its structure and number of nodes without explicitly enumerating all possible tree structures. The minimal length of the bit sequence makes it possible to optimize trees with the fixed nodes number using genetic algorithms

Keywords

Binary tree, Catalan numbers, encoding algorithm, minimal redundancy

REFERENCES

[1] Korvyakov V.P. Method of neuro-fuzzy estimation of graphical user interface usability. *Vestn. Mosk. Gos. Tekh. Univ. im. N.E. Baumana, Priborostr.* [Herald of the Bauman Moscow State Tech. Univ., Instrum. Eng.], 2016, no. 5, pp. 61–74 (in Russ.). DOI: 10.18698/0236-3933-2016-5-61-74

[2] Knuth D.E. The art of computer programming. Vol. 1. Fundamental Algorithms. Reading, Massachusetts: Addison-Wesley, 1997. 650 p.

[3] Davis T. Catalan numbers. geometer.org: website. Available at: http://www.geometer.org/mathcircles/catalan.pdf (accessed 17.08.2016).

[4] Bege A., Kasa Z. Coding objects related to Catalan numbers. *Studia Universitatis Babes-Bolyai. Informatica*, 2001, vol. 46, no. 1, pp. 31–40. Available at: http://www.cs.ubbcluj.ro/~studia-i/2001-1/3-Kasa.pdf

[5] Flajolet P., Sedgewick R. Analytic combinatorics. Cambridge University Press, 2009. 826 p.

[6] Farzan A., Munro J.I. A uniform paradigm to succinctly encode various families of trees. *Algorithmica*, 2014, vol. 68, no. 1, pp. 16–40. DOI: 10.1007/s00453-012-9664-0 Available at: http://link.springer.com/article/10.1007%2Fs00453-012-9664-0

[7] Davoodi P., Raman R., Satti S.R. On succinct representations of binary trees. arXiv.org: Cornell University Library. Available at: http://arxiv.org/abs/1410.4963 (accessed 29.09.2016).

[8] Mäkinen E. A survey on binary tree coding. *The Computer Journal*, 1991, vol. 34, no. 5, pp. 438–443. DOI: 10.1093/comjnl/34.5.438 Available at: https://academic.oup.com/comjnl/article-abstract/34/5/438/553944/A-Survey-on-Binary-Tree-Codings?redirectedFrom=fulltext

[9] Cormen T.H., Leiserson Ch.E., Rivest R.L., Stein C. Introduction to algorithms. MIT Press, 2009. 235 p.

[10] Catalantree — Binary trees decoding/encoding with Catalan numbers based algorithm. GitHub: website. Available at: https://github.com/HapKoM/catalantree (accessed 24.08.2016).

[11] Gansner E., Koutsofios E., North E. Drawing graphs with dot. Graphviz: website. Available at: http://www.graphviz.org/Documentation/dotguide.pdf (accessed 24.08.2016).

[12] Knuth D.E. The art of computer programming. Vol. 4. Fascicle 4: Generating all trees–history of combinatorial generation. Upper Saddle River, New Jersey, Addison-Wesley, 2011. 883 p.

Korvyakov V.P. — post-graduate student, software engineer Research and Development Centre, S.P. Korolev Rocket and Space Corporation Energia (Lenina ul. 4a, Korolev, Moscow Region, 141070 Russian Federation).

Please cite this article in English as:

Korvyakov V.P. An Algorithm of Binary Tree Encoding with Minimum Redundancy. *Vestn. Mosk. Gos. Tekh. Univ. im. N.E. Baumana, Priborostr.* [Herald of the Bauman Moscow State Tech. Univ., Instrum. Eng.], 2017, no. 3, pp. 33–46. DOI: 10.18698/0236-3933-2017-3-33-46