Вестник Ивановского государственного университета

Серия «Биология, Химия, Физика, Математика»

Вып. 3 / 2004 С. 139 – 143

А. А. Толстопятов, С. И. Хашин

АЛГОРИТМ ПОСТРОЕНИЯ ПОЛЯ ПОРЯДКА ПРИ БУЛЕВОМ СЖАТИИ

По данному типу перестановки вычисляется номер каждой конкретной перестановки. И наоборот, по данному типу и номеру находится конкретная перестановка.

By the given type of a permutation the number of every concrete permutation is calculated. Conversely, by the given type and the number every concrete permutation is obtained.

УДК 514.54.

Введение

Рассмотрим последовательность $R = \{x_0,\ \ldots, x_{m-1}\}$ из m целых чисел x_i, каждое из которых принадлежит отрезку $[0,\ 2^N - 1]$ для некоторого N. Элементы в последовательности могут повторяться. Рассмотрим множество чисел, встречающихся в данной последовательности. Обозначим их, в порядке возрастания, $j_0 < j_1 < \cdots < j_{s-1}$, где s — количество различных элементов в R. Через n_k, $k = 0,\ \ldots, s-1$, обозначим количество чисел в последовательности, равных j_s, при этом $n_0 + \cdots + n_{s-1} = m$. Набор чисел n_0, ... , n_{s-1} будем называть *типом повторов* [2].

Сформулируем теперь две задачи, которые рассматриваются в работе. Для обеих задач фиксируется тип повторов — некоторая последовательность натуральных чисел $P = \{n_0,\ \ldots, n_{s-1}\}$. Рассмотрим последовательности натуральных чисел $I = \{i_0,\ \ldots, i_{m-1}\}$, длины $m = n_0 + \cdots + n_{s-1}$, $i_k \in \{0,\ \ldots, s-1\}$ такие, что каждое натуральное число j встречается в последовательности n_j раз. Такие последовательности будем называть *соответствующими* типу повторов P. Их количество равно $m!/(n_0! \cdot n_1! \cdot \cdots \cdot n_{s-1}!)$. Множество таких последовательностей обозначим $S(P)$.

Например, выберем тип повторов, равный $P = \{1, 2, 2\}$, т. е. $s = 3$, $n_0 = 1$, $n_1 = n_2 = 2$, $m = n_0 + n_1 + n_2 = 5$. Рассматриваемые последовательности состоят из 5 натуральных чисел $I = \{i_0, i_1, i_2, i_3, i_4\}$, среди которых один ноль ($n_0 = 1$), две единицы ($n_1 = 2$), две двойки ($n_2 = 2$).

Работа выполнена при финансовой поддержке РФФИ (проект 02-01-000-79).

Таких последовательностей $30 = 5!/(1! \cdot 2! \cdot 2!)$ штук:

$$(01122),(10122),(11022),(11202),(11220),$$
$$(01212),(10212),(12012),(12102),(12120),$$
$$(01221),(10221),(12021),(12201),(12210),$$
$$(02112),(20112),(21012),(21102),(21120),$$
$$(02121),(20121),(21021),(21201),(21210),$$
$$(02211),(20211),(22011),(22101),(22110).$$

Задача 1. Для фиксированного типа повторов P и данной последовательности, относящейся к этому типу, найти ее порядковый номер в множестве $S(P)$.

Задача 2. Для фиксированного "типа повторов" P и заданного порядкового номера в множестве $S(P)$ найти соответствующую последовательность.

В обеих задачах предполагается некоторое упорядочение в множестве $S(P)$. Упорядочение может быть любым, но одинаковым для обеих задач.

2. Алгоритм обхода всех последовательностей фиксированного типа

Ключевым моментом для решения обеих задач является функция, которая по данному типу повторов P и последовательности, относящейся к этому типу, т. е. принадлежащей к $S(P)$, находит “следующую”последовательность из $S(P)$.

Определение. Пусть дана последовательность натуральных чисел $I = \{i_0, \ldots, i_{m-1}\}$, принадлежащая к типу $P = \{n_0, \ldots, n_{s-1}\}$, т. е. $n_k =$ количеству i_j, равных k. Назовем элемент i_k *активным уровня* n, если $i_k = n$, $i_{k+1} > i_k$ и индекс k наименьший среди обладающих указанным свойством.

Например, для последовательности (11022) активным уровня 0 будет элемент i_2, активным уровня 1 будет элемент i_1, активного элемента уровня 2 не существует.

Выберем такой алгоритм нахождения “следующей”последовательности.

1. Устанавливаем переменную l — уровень активности в 0.

2. Находим i_k — активный элемент уровня l.

3. Если такого не существует:

3а. Если l — максимальное, то сообщаем, что “следующей”последовательности не существует и заканчиваем работу.

3б. Увеличиваем l и переходим к шагу 2.

4. Активный элемент найден — i_k. Тогда:

5. Меняем в последовательности местами i_k и i_{k+1}.

6. Среди элементов подпоследовательности I, стоящих левее k и меньших либо равных l, все элементы, равные l переставляем в начало.

Построенную последовательность и объявляем “следующей”.

Примеры. 1. $Next(01122) = (10122)$; 2. $Next(11022) = (11202)$.

Номером подстановки при таком подходе назовем номер, под которым она возникает при полном переборе всех подстановок с фиксированным типом повторов, начиная с лексикографически минимальной.

3. Алгоритм посчета количества подстановок, лексикографически меньших данной

Обозначения. Так же, как и выше, для данной подстановки I через $P(I)$ обозначим ее тип повторов. Через $c(I)$ обозначим количество подстановок с тем же типом повторов, что и у I, т. е. мощность множества $S(P(I))$ [1]:

$$c(I) = m!/(n_0! \cdot n_1! \cdot \dots \cdot n_{s-1}!)$$

где $P(I) = \{n_0, \dots, n_{s-1}\}$. Для данного типа повторов $P = \{n_0, \dots, n_{s-1}\}$ через P_k обозначим тип повторов, в котором n_k уменьшено на 1. Через $c_k(I)$ обозначим мощность множества $P_k(I)$. Через $N(I)$ обозначим количество подстановок с тем же типом повторов, лексикографически меньших I.

Теорема. *Для номера подстановки $N(I) = N(\{i_0, \dots, i_{m-1}\})$ верна формула:*

$$N(I) = \sum_{i=0}^{i_0-1} c_i(I) + N(I'),$$

где $I' = \{i_1, \dots, i_{m-1}\}$ — подстановка I с выброшенным первым элементом.

Доказательство. Множество подстановок, лексикографически меньших I, можно представить в виде объединения двух подмножеств, состоящих из подстановок, у которых первый элемент меньше, чем у I, и подстановок, у которых первый элемент равен i.

Первое из указанных подмножеств разбивается на классы в соответствии со своим первым элементом $i \in \{0, \dots, i_0 - 1\}$, мощность каждого из классов равна $c_i(I)$, мощность второго подмножества равна как раз $N(I')$.

Так как для величин $c_i(I)$ имеется явная формула, то доказанная теорема дает достаточно эффективный (правда, рекуррентный) способ вычисления номера подстановки.

4. Реализация алгоритма

Для реализации описанных алгоритмов был выбран язык Java. Отметим основные черты языка, повлиявшие на этот выбор:

— открытость. Распространение java-программ вместе с исходными текстами является обычной практикой. Если даже их нет, java-классы легко декомпилируются;

— бесплатность. И компилятор, и виртуальная java-машины распространяются бесплатно [3];

— наличие класса, работающего с целыми числами произвольной длины, — BigInteger.

Среди методов класса BigInteger, помимо стандартных арифметических операций, выделим следующие:

— преобразование из десятичного и шестнадцатиричного представления и обратно;

— mod — взятие числа по модулю;

— modInvers — нахождение обратного по модулю;

— modPow — возведение в степень по модулю.

Реализация этих операций в языке тщательно оптимизирована. Так как все стандартные библиотеки языка Java поставляются вместе с исходными текстами, их реализацию можно подробно изучить (*src.zip/java /math/BigInteger.java*).

Для примера приведем время выполнения одной из наиболее сложных операций, *modPow*, на процессоре с тактовой частотой 1700 МГц в зависимости от разрядности чисел:

длина (байт)	длина (бит)	время (мс)
8	64	0.16
16	128	0.41
32	256	1.98
64	512	11.80
128	1024	83.20
256	2048	620.00

Для работы с классом BigInteger не требуется никаких дополнительных библиотек, что также является существенным фактором.

Алгоритмы, описанные выше были реализованы в классе Subst.

Подход, предложенный в п. 2, реализован в методах Subst.indexOf_1(), Subst.setByN_1(int k). Он дает максимально полную информацию о последовательности подстановок. С другой стороны, он требует значительных вычислительных ресурсов. Приведем время, требуемое для вычисления номера подстановки

$$J_k = \{k-2, k-2, k-3,\ \ldots, 1, 0\}.$$

k	время (с)
7	0.015
9	0.045
10	0.391
11	4.200
12	54.500

Фактически данный способ пригоден лишь для подстановок не длиннее 12 — 13 чисел.

Подход, описанный в п. 3, реализован в методе Subst.indexOf_2(). Приведем время, требуемое для вычисления номера подстановки

$$J_k = \{k-2, k-2, k-3,\ \ldots, 1, 0\}.$$

k	время (с)
7	0.015
10	0.016
20	0.031
50	0.140
100	0.812
200	9.400
300	45.000

Таким образом, данный способ пригоден лишь для подстановок длины вплоть до 300 чисел.

Библиографический список

1. *Толстопятов А. А.* О возможности использования булевых уравнений для сжатия файлов // Вестн. Иван. гос. ун-та. Биология. Химия. Физика. Математика. 2003. Вып. 4. С. 82—86.
2. *Толстопятов А. А.* О структуре дискретной информации и общих условиях ее сжатия // Там же. 2002. Вып. 3. С. 80—82.
3. Java(TM) 2 SDK, Standard Edition Version 1.4.2. http://java.sun.com.