Секция 7

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 004.021 DOI 10.17223/2226308X/10/57

К ЗАДАЧЕ О ЦИКЛИЧЕСКОЙ ПЕРЕСТАНОВКЕ ЭЛЕМЕНТОВ ОДНОМЕРНОГО МАССИВА

В. В. Гоцуленко

Получено выражение для наименьшего числа попарных перестановок элементов одномерного массива, за которое можно осуществить его циклический сдвиг на k позиций; построен алгоритм, осуществляющий k-циклическую перестановку.

Ключевые слова: *одномерный массив, k-циклическая перестановка, попарная перестановка элементов массива.*

Введение

Задача циклического сдвига одномерного массива из N элементов на k позиций вправо или влево является классической, казалось бы, ничем не выдающейся олимпиадной «задачкой». Например, если $N = 11$, $k = 3$ и для определённости сдвиг осуществляется вправо, то вектор «abracadabra» превратится в «braabracada».

Однако эта задача имеет важное прикладное значение. Например, во всех современных текстовых редакторах обязательно присутствует возможность выделения мышкой текста и последующего его перемещения в любое другое место редактируемого файла. Операция циклического сдвига в некоторых языках программирования является элементарной [1], то есть выполняется одним оператором. При этом важно, что циклический сдвиг соответствует обмену соседних блоков памяти разного размера. Ограничения по времени и объёму памяти существенны для многих подобных приложений.

Можно решить данную задачу, копируя первые k элементов массива во временный буфер, сдвигая оставшиеся $(N - k)$ элементов влево на k позиций, а затем копируя данные из буфера в основной массив на последние k позиций. Однако данная схема использует k дополнительных переменных, т. е. требует выделения дополнительной памяти. Можно определить функцию, сдвигающую массив влево на один элемент (за время, пропорциональное N), а потом вызвать её k раз, но такой алгоритм будет отнимать слишком много времени.

Другое очевидное решение «в лоб» — также использовать вспомогательный буферный N-элементный массив. Сначала элементы исходного массива перемещаются в буферный массив с сохранением их порядка. Далее в позицию с номером i $(0 \leqslant i < N)$ исходного массива вставляется элемент с позиции $f(i)$ из буферного массива, где отображение $f : \{0, \ldots, N-1\} \to \{0, \ldots, N-1\}$ задаётся по правилу

$$f(i) = \begin{cases} i - k, \text{ если } i \geqslant k, \\ N\,(1 + [k/N]) - k + i \text{ иначе,} \end{cases} \tag{1}$$

где через $[x]$ обозначена целая часть вещественного числа x. В этом решении мы также сталкиваемся с дополнительным расходом памяти.

Таким образом, актуальным является построение алгоритма, который осуществляет перестановку элементов исходного массива, по возможности с наименьшим их перемещением в памяти и без дополнительного её расхода, или, по-крайней мере, чтобы этот расход не рос пропорционально объёму сдвигаемого фрагмента данных.

1. Алгоритмы циклической перестановки элементов массива

Существует несколько проверенных временем и опытом алгоритмов [1, 2], которые позволяют обойтись лишь несколькими дополнительными переменными и завершить весь сдвиг за время, пропорциональное N.

Алгоритм A. Вместо использования буферного массива введём только одну дополнительную переменную. Помещаем первый элемент массива $x[0]$ во временную переменную t. Оставшиеся $N-1$ элементов массива перемещаются на одну позицию влево, и в конец массива помещается содержимое переменной t. После применения данной процедуры $m = k \bmod N$ раз получаем необходимую перестановку. Несложно подсчитать, что реализация такого алгоритма использует $m\,(N+1)$ операций перемещения элементов массива. Сложность алгоритма можно существенно уменьшить, если вместо m-разового применения операции циклического сдвига на одну позицию каждый элемент сразу сдвигать на m позиций. Таким образом приходим к следующему алгоритму.

Алгоритм B. Элемент $x[0]$ помещается во временную переменную t, затем $x[m]$ помещается в $x[0]$, $x[2m]$ — в $x[m]$ и так далее (перебираются все элементы массива x с одинаковыми по модулю N индексами), пока не возвращаемся к элементу $x[0]$, вместо которого записывается содержимое переменной t. Если при этом не были переставлены все элементы, процедура повторяется, начиная с $x[1]$, и так далее до достижения конечного результата.

Отметим, что сложность данного алгоритма в сравнении с предыдущим примерно в m раз меньше.

Алгоритм C. Можно предложить другой алгоритм, который возникает из рассмотрения задачи с другой точки зрения. Циклический сдвиг массива x сводится фактически к замене $\alpha\beta$ на $\beta\alpha$, где α — первые k элементов массива x, β — оставшиеся элементы. Для определённости предположим, что α короче β. Тогда разобьём β на β_{left} и β_{right}, где β_{right} содержит k элементов (столько же, сколько и α). Поменяем местами α и β_{right}, получим $\beta_{\text{right}}\beta_{\text{left}}\alpha$. При этом α окажется в конце массива — там, где и полагается. Поэтому можно сосредоточиться на перестановке β_{right} и β_{left}. Однако эта задача сводится к начальной, поэтому алгоритм можно вызывать рекурсивно.

Алгоритм D. Нужно преобразовать массив $\{\alpha\beta\}$ в $\{\beta\alpha\}$. Предположим, что у нас есть функция «переворота» reverse, переставляющая элементы массива в противоположном порядке. В исходном состоянии массив имеет вид $\alpha\beta$. Алгоритм состоит в трёхразовом применении функции reverse: $\beta\alpha = \text{reverse}\,(\text{reverse}\,(\alpha)\,\text{reverse}\,(\beta))$.

В [1] приведено: «Код, использующий функцию переворота, оказывается эффективным и малотребовательным к памяти, и настолько короток и прост, что при его реализации сложно ошибиться. Б. Керниган и П. Дж. Плоджер пользовались именно этим методом для перемещения строк в текстовом редакторе. В [3] Керниган пишет: «Эта функция заработала правильно с первого же запуска, тогда как их предыдущая версия, использовавшая связный список, содержала несколько ошибок. Этот же код используется в некоторых текстовых редакторах, включая тот, в котором я впервые

набрал настоящую главу. Кен Томпсон (Ken Thompson) написал этот редактор с функцией reverse в 1971 году, и он утверждает, что она уже тогда была легендарной».

Ниже предложен новый алгоритм циклической перестановки элементов одномерного массива, который вообще не требует дополнительной памяти и использует минимально возможное число попарных перестановок элементов.

2. Формулировка основного результата

Для произвольного $0 \leqslant i < N$ обозначим через $\omega(i)$ наименьшее целое число, для которого выполняется неравенство

$$f^{(\omega(i))}(i) \equiv f(f(\ldots f(i))) \geqslant i,$$

где операция композиции выполняется $\omega(i)$ раз. Отметим, что число $\omega(i)$ существует. Действительно, среди $i+1$ чисел $f^{(j)}(i)$, $j = 0, 1, \ldots, i$, по крайней мере одно не меньше i. Следовательно, отображение $\omega : \{0, \ldots, N-1\} \to \{0, \ldots, N-1\}$ определено корректно.

Теорема 1. Наименьшее число попарных перестановок элементов одномерного массива $(\alpha[0], \alpha[1], \ldots, \alpha[N-1])$, за которое можно осуществить его циклический сдвиг на k позиций вправо, равно мощности $J(N,k)$ множества

$$\{i : N > i \geqslant 0 \ \&\ g(i) > i\},$$

где отображение $g : \{0, \ldots, N-1\} \to \{0, \ldots, N-1\}$ задаётся правилом

$$g(i) = f^{(\omega(i))}(i) \quad (0 \leqslant i < N), \tag{2}$$

а отображение f определяется соотношением (1).

Не нарушая общности, ограничимся рассмотрением случая $0 < k < N$. Для отображения (2) можно получить явное представление. Действительно, в случае $0 \leqslant i < k$ из (1) следует, что $f(i) > i$ и, следовательно, $\omega(i) = 0$. В случае $i \geqslant k$ положим $i = k + j$, где $0 \leqslant j < N - k$. Непосредственно проверяется, что для всех $\omega \geqslant 0$

$$f^{(\omega)}(k+j) = rN - qk + j, \text{ где } 0 \leqslant r \leqslant q, \ q \geqslant 1.$$

Таким образом, задача определения наименьшего показателя композиции ω, при котором выполняется неравенство $k+j \leqslant f^{(\omega)}(k+j) \leqslant N-1$, приводит к следующей задаче целочисленной минимизации: определить точку $P(r,q)$ на плоскости с наименьшими целыми координатами r, q, которая расположена в области, заданной неравенствами

$$\frac{k}{N} \leqslant r - \frac{k}{N}q \leqslant \frac{N-j-1}{N}, \ 1 \leqslant r \leqslant q, \ q \geqslant 1. \tag{3}$$

Перейдём непосредственно к доказательству теоремы. Введём в рассмотрение операцию $\mathrm{swap}(i,j)$ перестановки элементов массива с индексами i и j. Для реализации данной операции не обязательно использовать вспомогательную буферную переменную, например, если на множестве элементов рассматриваемого массива α определены взаимно обратные операции сложения и вычитания, то можно обменять содержимое элементов $\alpha[i]$ и $\alpha[j]$, выполнив последовательно следующие операции: 1) $\alpha[i] := \alpha[i] + \alpha[j]$; 2) $\alpha[j] := \alpha[i] - \alpha[j]$; 3) $\alpha[i] := \alpha[i] - \alpha[j]$.

Далее, будем последовательно выполнять $\mathrm{swap}\,(i, f\,(i))$ $(0 \leqslant i < N)$ до тех пор, пока выполняется неравенство $f\,(i) > i$. Если для текущего значения i выполняется противоположное неравенство $f\,(i) < i$, то выполнение операции $\mathrm{swap}\,(i, f\,(i))$ приводит к тому, что элемент $\alpha\,[i]$ обменивается содержимым с элементом $\alpha\,[f\,(i)]$, который ранее уже подвергался выполнению операции swap. Следовательно, в этом случае для определения правильного значения индекса второго элемента необходимо использовать функцию (2). Окончательно рассматриваемый алгоритм циклической перестановки может быть сформулирован в следующей форме.

Алгоритм E. Циклический сдвиг вправо на k позиций произвольного N-элементного массива $(\alpha\,[0]\,, \alpha\,[1]\,, \ldots, \alpha\,[N-1])$ можно осуществить, выполняя $J\,(N, k)$ последовательных попарных перестановок его элементов:

$$\mathrm{swap}\,(0, g\,(0)) \circ \mathrm{swap}\,(1, g\,(1)) \circ \ldots \circ \mathrm{swap}\,(J\,(N, k) - 1, g\,(J\,(N, k) - 1))\,.$$

В таблице приведены некоторые примеры применения данного алгоритма. В теоретико-групповой трактовке [4] полученный результат можно сформулировать следующим образом: любая k-циклическая перестановка элементов произвольного N-элементного массива может быть представлена как композиция не более чем $J\,(N, k)$ транспозиций его элементов.

$k = 1,\ N = 6$			$k = 2,\ N = 6$		
([0] 1 2 3 4 [5])	$i \to g(i)$	$i \to f(i)$	([0] 1 2 3 [4] 5)	$i \to g(i)$	$i \to f(i)$
(5 [1] 2 3 4 [0])	$0 \to 5$	$0 \to 5$	(4 [1] 2 3 0 [5])	$0 \to 4$	$0 \to 4$
(5 0 [2] 3 4 [1])	$1 \to 0 \to 5$	$1 \to 0$	(4 5 [2] 3 [0] 1)	$1 \to 5$	$1 \to 5$
(5 0 1 [3] 4 [2])	$2 \to 1 \to 5$	$2 \to 1$	(4 5 0 [3] 2 [1])	$2 \to 0 \to 4$	$2 \to 0$
(5 0 1 2 [4] [3])	$3 \to 2 \to 5$	$3 \to 2$	[(4 5 0 1 2 3)]	$3 \to 1 \to 5$	$3 \to 1$
[(5 0 1 2 3 4)]	$4 \to 3 \to 5$	$4 \to 3$		$4 \to 2 \to 4$	$4 \to 2$
	$5 \to 4 \to 5$	$5 \to 4$		$5 \to 3 \to 5$	$5 \to 3$
$k = 3,\ N = 6$			$k = 4,\ N = 6$		
([0] 1 2 [3] 4 5)	$i \to g(i)$	$i \to f(i)$	([0] 1 [2] 3 4 2)	$i \to g(i)$	$i \to f(i)$
(3 [1] 2 0 [4] 5)	$0 \to 3$	$0 \to 3$	(2 [1] 0 [3] 4 5)	$0 \to 2$	$0 \to 2$
(3 4 [2] 0 1 [5])	$1 \to 4$	$1 \to 4$	(2 3 [0] 1 [4] 5)	$1 \to 3$	$1 \to 3$
[(3 4 5 0 1 2)]	$2 \to 5$	$2 \to 5$	(2 3 4 [1] 0 [5])	$2 \to 4$	$2 \to 4$
	$3 \to 0 \to 3$	$3 \to 0$	[(2 3 4 5 0 1)]	$3 \to 5$	$3 \to 5$
	$4 \to 1 \to 4$	$4 \to 1$		$4 \to 0 \to 2 \to 4$	$4 \to 0$
	$5 \to 2 \to 5$	$5 \to 2$		$5 \to 1 \to 3 \to 5$	$5 \to 1$

$k = 5,\ N = 6$		
([0] [1] 2 3 4 5)	$i \to g(i)$	$i \to f(i)$
(1 [0] [2] 3 4 5)	$0 \to 1$	$0 \to 1$
(1 2 [0] [3] 4 5)	$1 \to 2$	$1 \to 2$
(1 2 3 [0] [4] 5)	$2 \to 3$	$2 \to 3$
(1 2 3 4 [0] [5])	$3 \to 4$	$3 \to 4$
[(1 2 3 4 5 0)]	$4 \to 5$	$4 \to 5$
	$5 \to 0 \to 1 \to 2 \to 3 \to 4 \to 5$	$5 \to 0$

Заключение

Эффективность реализации алгоритма E определяется способом вычисления функции (2). С геометрической точки зрения эта задача сводится к решению задачи цело-

численной минимизации (3). В качестве дальнейшего предмета исследования является открытым вопрос построения, по возможности, явного решения задачи (3).

ЛИТЕРАТУРА

1. *Бентли Дж.* Жемчужины программирования. СПб.: Питер, 2002. 272 с.
2. *Столяр С. Е.* Массивы. СПб.: ЦПО «Информатизация образования», 2002. 39 с.
3. *Kernighan B. and Plauger P. J.* Software Tools in Pascal. Boston: Addison-Wesley, 1981.
4. *Холл М.* Теория групп. М.: ИЛ, 1962. 460 с.

УДК 519.682 DOI 10.17223/2226308X/10/58

АНАЛОГ ТЕОРЕМЫ О НЕЯВНОМ ОТОБРАЖЕНИИ ДЛЯ ФОРМАЛЬНЫХ ГРАММАТИК

О. И. Егорушкин, И. В. Колбасина, К. В. Сафонов

В работе продолжается исследование систем некоммутативных полиномиальных уравнений, которые интерпретируются как грамматики формальных языков. Такие системы решаются в виде формальных степенных рядов (ФСР), выражающих нетерминальные символы через терминальные символы алфавита и рассматриваемых как формальные языки. Всякому ФСР поставлен в соответствие его коммутативный образ, который получается в предположении, что все символы обозначают коммутативные переменные, принимающие значения из поля комплексных чисел. В продолжение исследований совместности систем некоммутативных полиномиальных уравнений, которая напрямую не связана с совместностью её коммутативного образа, получено достаточное условие совместности в виде аналога теоремы о неявном отображении для формальных грамматик: если для коммутативного образа системы выполнено условие теоремы о неявном отображении, то не только она, но и исходная система некоммутативных уравнений имеет единственное решение в виде ФСР.

Ключевые слова: *системы полиномиальных уравнений, некоммутативные переменные, формальный степенной ряд, коммутативный образ, якобиан.*

Продолжая исследование, начатое в работах [1, 2], рассмотрим систему полиномиальных уравнений

$$P_j(z, x) = 0, \quad P_j(0, 0) = 0, \quad j = 1, \ldots, k, \tag{1}$$

которая решается относительно символов $z = (z_1, \ldots, z_n)$ в виде ФСР, зависящих от символов $x = (x_1, \ldots, x_m)$.

Такие системы имеют приложения в теории формальных языков, поскольку являются грамматиками, порождающими важные классы формальных языков: контекстно-свободных, языков непосредственно составляющих, языков в нормальной форме Грейбах и др. [3, 4].

В теории формальных языков символы $x_1, \ldots, x_m$ называются терминальными и образуют словарь (алфавит) данного языка, а символы $z_1, \ldots, z_n$ называются нетерминальными и необходимы для задания грамматических правил. Над всеми символами определены некоммутативная операция умножения (конкатенации), коммутативная операция формальной суммы, а также коммутативная операция умножения на комплексные числа, поэтому можно рассматривать символьные многочлены и ФСР с числовыми (комплексными) коэффициентами. Наконец, мономы от терминальных символов интерпретируются как предложения языка, а каждый ФСР, который является