Компьютерные инструменты в образовании, 2020
№ 1: 5–18
УДК: 512.12
http://cte.eltech.ru
doi:10.32603/2071-2340-2020-1-5-18

алгоритмическая математика и математическое моделирование

МЕТОД СОКРАЩЕНИЯ ПЕРЕБОРА В АЛГОРИТМАХ ПОСТРОЕНИЯ МИНИМАЛЬНЫХ АДДИТИВНЫХ ЦЕПОЧЕК

Коточигов А. М.[1], доктор физико-математических наук, профессор,
amkotochigov@gmail.com
Сучков А. И.[1], аспирант, suchkov3381@gmail.com

[1]Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина), ул. Профессора Попова, д. 5, корп. 3, 197376, Санкт-Петербург, Россия

Аннотация

Оптимизация алгоритмов вычислений значений многочленов, точнее мономов, эквивалентна задаче построения для заданного числа минимальной аддитивной цепочки. Для поиска таких цепочек не известно никаких алгоритмов, кроме перебора. Рост сложности перебора очень велик. Среди цепочек одинаковой длины очень много эквивалентных, то есть заканчивающихся одинаковым числом.

В статье приведен достаточный признак эквивалентности цепочек и показано, как использование признака позволяет сократить процедуру формирования всех аддитивных цепочек фиксированной длины.

Ключевые слова: *аддитивные цепочки, эквивалентные цепочки, пассивный интервал, существенное перемешивание.*

Цитирование: Коточигов А. М., Сучков А. И. Метод сокращения перебора в алгоритмах построения минимальных аддитивных цепочек // Компьютерные инструменты в образовании. 2020. № 1. С. 5–18. doi:10.32603/2071-2340-2020-1-5-18

1. ВВЕДЕНИЕ

Если вам нужно вычислить 2^{2019}, то не стоит 2018 раз умножать числа на 2. Значительно быстрее вычислить за 10 умножений числа

$$2^2,\ 2^4,\ 2^8,\ 2^{16},\ 2^{32},\ 2^{64},\ 2^{128},\ 2^{256},\ 2^{512},2^{1024},$$

затем за 7 умножений получить ответ

$$2^{1024}2^{512},$$
$$2^{1024}2^{512}2^{256},$$
$$2^{1024}2^{512}2^{256}2^{128},$$
$$2^{1024}2^{512}2^{256}2^{128}2^{64},$$
$$2^{1024}2^{512}2^{256}2^{128}2^{64}2^{32},$$
$$2^{1024}2^{512}2^{256}2^{128}2^{64}2^{32}2^2,$$
$$2^{1024}2^{512}2^{256}2^{128}2^{64}2^{32}2^2 2 = 2^{2019}.$$

Эти длинные записи являются следствием тождества

$$2^{2019} = 2^{1024+512+256+128+64+32+2+1} =$$

$$= 2 \times 2^2 \times 2^{32} \times 2^{64} \times 2^{128} \times 2^{512} \times 2^{1024}.$$

Из этого наблюдения ясно, что количество операций умножения, необходимых для вычисления числа можно оптимизировать, если воспользоваться структурой диадического разложения числа. Современное состояние вычислительной техники позволяет создавать модели, использующие многочлены высоких степеней от многих переменных. Для вычисления выражения вида

$$x^{123} y^{4321} z^{12345}$$

оптимизация алгоритма весьма актуальна.

Вопрос об оптимальном алгоритме вычисления степеней до сих пор остается открытым, поскольку алгоритм, основанный на диадическом разложении, не всегда оптимален. Например, для возведения числа в 15 степень по «диадическому» алгоритму требуется 6 шагов, но можно провести вычисление и за 5 шагов:

1	2	4	8	12	14	15
1	2	3	5	10	15	

Из приведенных примеров видно, что эта задача эквивалентна поиску для заданного числа N последовательности чисел

$$c_1 = 1,\ c_k = c_{k_1} + c_{k_2},\ 1 \le k_1, k_2 < k,\ k = 2, 3, \ldots, m.$$

Такие последовательности называются *аддитивными цепочками*.

Аддитивную цепочку, представляющую заданное число ($c_m = N$) и имеющую наименьшую возможную длину, называют *минимальной* аддитивной цепочкой для этого числа. Минимальная аддитивная цепочка позволяет вычислить степень числа за минимальное число умножений. Из определения следует, что аддитивных цепочек длины n существует

$$((n-1)!)^2 \sim \left(\frac{n}{e}\right)^{2n}.$$

Поиск цепочки путем перебора оказывается очень затратным.

2. ОБЗОР ЛИТЕРАТУРЫ

Аддитивные цепочки интенсивно изучаются, но ни одно из известных их свойств не дает оснований для формирования универсального алгоритма поиска минимальных цепочек. Терминология, используемая для описания аддитивных цепочек, и некоторые свойства аддитивных цепочек изложены в монографии Д. Кнута [1]. Например, некоторое время существовала гипотеза о том, что минимальную цепочку всегда можно найти в виде, содержащем на один произвольный параметр меньше

$$c_1 = 1, c_k = c_{k-1} + c_{r_k}, 1 \le r_k < k,\ k = 2, 3, \ldots, m.$$

Такие цепочки называют *звездными цепочками*. Гипотеза оказалась неверной: для числа 12509 минимальная аддитивная цепочка короче, чем минимальная звездная цепочка [1]. Тем не менее звездные цепочки остаются привлекательным объектом для исследований, хотя бы как промежуточный этап в решении задачи в полном объёме. В настоящее время активно исследуются вопросы связанные с вычислением выражений вида

$x^A y^B z^C$. Здесь нет универсальных алгоритмов, но описаны алгоритмы параллельного построения аддитивных цепочек для нескольких оснований. Обзор таких алгоритмов имеется в статье [2]. В статье [3] изложен матричный подход к описанию таких алгоритмов, основанный на формальном описании процесса в терминах линейной алгебры, он позволяет взглянуть на такие алгоритмы с общих позиций.

3. ЭКВИВАЛЕНТНЫЕ ЦЕПОЧКИ

Целью статьи является описание свойств аддитивных цепочек, позволяющих сократить время их поиска. Основанием рассматриваемой здесь конструкции служит то обстоятельство, что звездных цепочек длины m существует $(m-1)!$, а все возможные конечные элементы c_m таких цепочек подчинены неравенству $m < c_m \le 2^{m-1}$. Следовательно, в среднем $\left(\dfrac{m-1}{2e}\right)^{m-1}$ цепочек имеют одинаковый последний элемент. Если появится возможность выделять цепочки, имеющие равный последний элемент, то процедура перебора могла бы ускориться. Цепочки одинаковой длины, имеющие равные последние элементы будем называть *эквивалентными цепочками*. В статье рассматриваются звездные цепочки, авторы рассматривают эту работу как подготовительный этап для перехода к аддитивным цепочкам общего вида.

Рассмотрим простейшую ситуацию — в паре цепочек длины m совпадают все элементы, кроме двух последних:

$$c_k = d_k,\ 1 \le k < m-1,\ c_{m-1} = c_{m-2} + c_{r_{m-1}},\ c_m = c_{m-1} + c_{r_m},$$

$$d_{m-1} = d_{m-2} + d_{\rho_{m-1}},\ d_m = d_{m-1} + d_{\rho_m},$$

причем $\rho_m = r_{m-1},\ \rho_{m-1} = r_m$. Это возможно только при условии, что $r_m < m-1$. Если это так, то

$$d_m = d_{m-1} + d_{\rho_m} = (d_{m-2} + d_{\rho_{m-1}}) + d_{\rho_m} = c_{m-2} + c_{r_m} + c_{r_{m-1}} = c_m,$$

то есть цепочки эквивалентны.

Для того чтобы обобщить это наблюдение, введем еще один термин. Будем говорить, что элемент цепочки c_k *пассивный*, если при любом j, большем k, выполнено условие $c_j = c_{j-1} + c_{r_j}, r_j \ne k$, то есть на него нет ссылок.

В этих терминах доказанное выше утверждение об эквивалентности цепочек можно сформулировать так: если элемент c_{m-1} пассивный, то перестановка «добавок» к элементам $m-1$ и m порождает эквивалентную цепочку.

Точно так же проверяется, что если на элемент c_k нет ссылок, то цепочка

$$d_j = c_j,\ j < k,\ d_k = d_{k-1} + d_{\rho_k},\ d_{k+1} = d_k + d_{\rho_{k+1}},$$

$$\rho_k = r_{k+1},\ \rho_{k+1} = r_k,\ \rho_j = r_j,\ j > k$$

эквивалентна цепочке $c_1, \ldots, c_m$. Нетрудно видеть, что $r_{k+1} < k$, и тогда $d_{k+1} = c_{k+1}$, а дальше ничего не меняется, так как на элемент c_k нет ссылок, а другие элементы не изменились. Отметим, что у любой цепочки хотя бы два элемента не являются пассивными: это первый элемент цепочки c_1, $c_2 = c_1 + c_1$ и последний элемент цепочки c_m (на него не может быть ссылок).

Поскольку описание пассивного элемента происходит в терминах индексов, то удобно называть *набором индексов* звездной цепочки $c_k = c_{k-1} + c_{r_k},\ k = 2, 3, \ldots, m$, множество

$(r_1, r_2, \ldots, r_m)$ (здесь $r_1 = 0$, символ r_1 введен ради унификации). Теперь можно сказать, что элемент c_k является пассивным тогда и только тогда, когда k не входит в множество индексов.

Очевидно, что звездные цепочки и наборы индексов находятся во взаимно однозначном соответствии.

Последовательность элементов $c_{p+1}, c_{p+2}, \ldots, c_{p+q}$ образует *пассивный интервал*, если каждый из ее элементов, кроме последнего, является пассивным.

3.1. Достаточное условие эквивалентности пары цепочек

Если в множестве индексов для каждого пассивного интервала провести произвольную перестановку относящихся к нему индексов, то новый набор индексов породит эквивалентную цепочку.

Наглядное представление о пассивном интервале можно получить, если изобразить цепочку как последовательность точек на прямой и соединить дугами точки, отвечающие позициям c_{r_k} и c_k. Тогда точки, из которых не выходит вправо ни одной дуги, являются пассивными.

Полезное представление звездной цепочки дает *матрица инцидентности* $(\mu_{k,j})_{1\le k,j\le m}$, которая определяется следующим образом:

$$\text{если } c_k = c_{k-1} + c_{r_k}, \text{ то } \mu_{k,r_k} = 1,\ \mu_{k,k-1} = 1, \text{ иначе } \mu_{k,j} = 0;$$

$$\text{если } c_k = c_{k-1} + c_{k-1}, \text{ то } \ \mu_{k,k-1} = 2, \text{ иначе } \mu_{k,j} = 0.$$

Элемент цепочки c_k является пассивным тогда и только тогда, когда

$$\sum_{1\le j\le m} \mu_{j,k} = 1.$$

Если элементы $c_{p+1}, c_{p+2}, \ldots, c_{p+q}$ образуют пассивный интервал, то перестановке индексов в этом интервале отвечает перестановка строк в блоке матрицы инцидентности

$$\begin{pmatrix} \mu_{p+1,1} & \cdots & \mu_{p+1,p-1} \\ \vdots & \vdots & \vdots \\ \mu_{p+q,1} & \cdots & \mu_{p+q,p-1} \end{pmatrix}.$$

Каждой такой перестановке отвечает звездная цепочка, эквивалентная исходной. У таких цепочек $c_k = d_k$ при $k \le p,\ \ k \ge p+q+1$.

Если у цепочки имеется несколько пассивных интервалов, то такую операцию можно проделать в каждом из них. Блоки, в которых происходят перестановки у разных интервалов, не пересекаются. Операцию по перестановке индексов в нескольких пассивных интервалах будем называть *перемешиванием*.

Множество цепочек, у которых позиции пассивных интервалов совпадают, а индексы в них могут отличаться друг от друга только перестановкой, будем называть *классом связанных цепочек*. Эти классы образуют дизъюнктное разбиение множества всех цепочек. Каждый класс порождается любой из входящих в него цепочек, все такие цепочки эквивалентны. Но, как показывает эксперимент, эквивалентными оказываются и некоторые другие цепочки.

Если у цепочки имеется s пассивных интервалов с длинами $q_1, \ldots, q_s$, то число элементов соответствующего класса связанных цепочек равно

$$(q_1)! \times (q_2)! \times \cdots \times (q_s)!.$$

3.2. Алгоритм перебора цепочек

Здесь будут рассматриваться цепочки фиксированной длины m. Алгоритм, приведенный ниже, описывает классы связанных цепочек. В результате работы алгоритма формируется список $List$ длины 2^{m-1} такой, что, если существует цепочка длины $c_1, \ldots, c_m$, последний элемент которой равен $c_m = n$, то

$$List(n) = (c_k), \text{ иначе } List(n) = 0,$$

при этом фиксируется только первое появление цепочки с конечным элементом $c_m = n$.

Параллельно формируется список $ListW$ длины 2^{n-1} такой, что по завершении работы $ListW(n)$ содержит список номеров цепочек (по одному из каждого класса связанных цепочек), у которых последний элемент равен $c_m = n$. Кроме того формируется список признаков $ListPrizn$ длины $(n-1)!$, в котором каждой цепочке сопоставлен номер класса связанных цепочек, которому она принадлежит.

Для реализации этого плана используется процедура нумерации цепочек, позволяющая по номеру восстановить цепочку и обратно по цепочке восстановить ее номер. Ввиду важности этой нумерации для реализации алгоритма в приложении дано описание такой нумерации.

3.3. Структура программы перебора

ЦИКЛ-1

В цикле меняется номер цепочки n от 1 до $(n-1)!$, по списку $ListPrizn$ проверяем, не попала ли эта цепочка в класс эквивалентных цепочек, рассмотренных ранее. Если «да», то переходим к следующему номеру (основной момент экономии). Если «нет», то число n преобразуется в соответствующий набор индексов $(r_1, \ldots, r_m)$, и цепочка восстанавливается.

Если в списке $List$ позиция с номером c_m не заполнена, то заносим туда цепочку и в такую же позицию списка $ListW$ заносим номер этой цепочки. Если в списке $List$ позиция с номером c_m уже заполнена, то в позицию c_m списка $ListW$ добавляем очередной номер цепочки.

Формируем список пассивных интервалов цепочки

$$(r_{p_k+1}, \ldots, r_{p_k+q_k}), \quad k = 1, \ldots, s,$$

и вычисляем общее число перемешиваний (всех возможных комбинаций перестановок на каждом пассивном интервале

$$T = (q_1)! \times (q_2)! \times \cdots \times (q_s)!.$$

ЦИКЛ-2

Формируем новый набор индексов для очередной эквивалентной цепочки. В цикле меняется номер перемешивания t от 1 до T. Используя процедуру нумерации, переводим номер перемешивания t в номера перестановок в каждом из пассивных интервалов $(t_1, \ldots, t_s)$.

ЦИКЛ-3

по всем пассивным интервалам k от 1 до s. Используя процедуру нумерации, переводим номер перестановки в саму перестановку.

Конец ЦИКЛА-3

Конец ЦИКЛА-2

В результате работы цикла-2 сформирован новый набор индексов (перестановки в пассивных интервалах). Процедура нумерации позволяет по набору индексов восстановить номер цепочки. Помечаем этот номер в списке $ListPrizn$, записывая туда текущий номер класса связанных цепочек.

Переходим в цикле-1 к следующему номеру цепочки.

Конец ЦИКЛА-1

В итоге получаем полную информацию о структуре цепочек: $List$ показывает, какие числа могут появляться в конце звездной цепочки длины m, $ListPrizn$ показывает, сколько существует различных классов цепочек, отличающихся только перемешиванием, и каков размер этих классов, $ListW$ содержит перечисление цепочек, являющихся эквивалентными, но не получающихся путем перемешивания.

3.4. Модифицированный алгоритм перебора цепочек

Рассмотренный алгоритм можно еще ускорить. Рассмотрим его работу с цепочкой

$$c_1 = 1, c_2 = 2, \ldots, c_n = n,$$

в которой $\mu_{2,1} = 2$, $\mu_{k,1} = 1$, $\mu_{k,k-1} = 1$, $k = 2, \ldots, n$, а остальные $\mu_{k,j} = 0$. Элементы $c_2, \ldots, c_{n-1}$ образуют пассивный интервал и элементы первого столбца матрицы инцидентности $\mu_{2,1}, \ldots, \mu_{n,1}$ можно как угодно переставлять. Но все эти элементы равны 1, и перестановки ничего не меняют. Эти «холостые» ходы можно устранить. Будем называть перестановку индексов пассивного интервала *существенной*, если в результате перестановки появится хотя бы одна позиция, где индекс изменился, то есть перестановка местами двух равных индексов не является существенной.

Структуру существенных перестановок в пассивном интервале несложно описать. Допустим, пассивный интервал имеет длину q и состоит из v групп равных индексов, содержащих $d_1, \ldots, d_v$ элементов ($d_1 + \cdots + d_v = q$). Тогда существенные перестановки можно перечислить так:

- на первом этапе берется произвольная выборка d_1-й позиции из q, всего таких выборов $C_q^{d_1}$; элементы первой группы ставятся на эти позиции, при этом $q - d_1$ позиций останутся свободными;
- на втором этапе нужно выбрать d_2 позиции из $q - d_1$, заполнить их элементами второй группы, всего таких выборов $C_{q-d_1}^{d_2}$;
- …;
- на последнем v-ом этапе остается заполнить d_v оставшихся позиций элементами последней группы.

Общее число существенных перестановок в пассивном интервале

$$C_q^{d_1} \times C_{q-d_1}^{d_2} \times C_{q-d_1-d_2}^{d_3} \cdots \times C_{d_v}^{d_v} = \frac{q}{d_1! d_2! \ldots d_v!}.$$

Если цепочка содержит несколько пассивных интервалов с длинами $q^{(j)}$, $j = 1, \ldots, s$, то проведем для каждого интервала описание структуры повторяющихся индексов

$$d_1^{(j)}, \ldots, d_{v_j}^{(j)}, \quad d_1^{(j)} + \cdots + d_v^{(j)} = q^{(j)}.$$

Естественным образом появляется термин *существенное перемешивание*. Общее число существенных перемешиваний

$$T = \prod_{1}^{s} \frac{q^{(j)}}{d_1^{(j)}! d_2^{(j)}! \dots d_\nu^{(j)}!}.$$

Для организации перебора всех существенных перемешиваний необходимо их перенумеровать. Преобразование номера существенного перемешивания в само перемешивание происходит так:

— на первом этапе по стандартной схеме, описанной в приложении, номер перемешивания переводится в набор номеров элементов прямого произведения множеств (существенных перестановок в каждом из пассивных интервалов),
— на втором этапе номер существенной перестановки в пассивном интервале интерпретируется как набор номеров выборок по d_j элементов из $q - \sum_{i=1}^{j-1} d_i$ (в прямом произведении наборов выборок для каждой группы одинаковых индексов),
— на третьем этапе номер выборки преобразуется в выборку (по алгоритму биномиального разложения, изложенному в [5]),
— на четвертом этапе эти выборки «вписываются» в пассивный интервал.

Таким образом, перечисляются все эквивалентные цепочки, а холостые ходы устраняются.

Приложение

Правило нумерации элементов прямого произведения

Пусть заданы множества G_r, $r = 1,2,\dots,s$, элементы которых занумерованы $G_r = (1,2,\dots,a_r)$. Рассмотрим граф, вершины которого распределены по уровням от 1 до s. На первом уровне располагается a_1 вершин (элементы первого множества). Из каждой вершины выходит a_2 ребра. Свободные концы этих ребер образуют $a_1 a_2$ вершин второго уровня (элементы прямого произведения $G_1 \times G_2$).

Продолжая процесс на уровне s получим $a_1 \times a_2 \times \dots \times a_s$ вершин (элементы прямого произведения $G_1 \times G_2 \times \dots \times G_s$).

Назовем *локальным номером* вершины на уровне j порядковый номер ребра, соединяющего ее с вершиной верхнего уровня (движение слева направо). *Глобальным номером* вершины на уроне j назовем ее порядковый номер при движении слева направо вдоль всего уровня. На s-ом уровне возникнет нумерация всех вершин прямого произведения.

Чтобы перейти от номера элемента прямого произведения к номерам элементов в каждом из множеств, достаточно подняться по ребрам с нижнего уровня на верхний. Локальные номера вершин $(n_1,\dots,n_s)$, образующих этот путь, и есть номера элементов в прямом произведении, отвечающие номеру элемента в нижнем уровне.

В ходе реализации алгоритма необходимо осуществлять переход к следующему глобальному номеру. Для того чтобы проводить эту процедуру в терминах локальных номеров, удобно ввести понятие *опорной вершины* для вершины на нижнем уровне. Пусть m $\left(1 \le m \le \prod_{k=1}^{s} a_k\right)$ — это глобальный номер вершины на нижнем уровне, $m \sim (m_1,\dots,m_s)$ — набор локальных номеров вершины m. Вершину m_j из списка локальных номеров (вершин) назовем опорной, если она первая снизу, имеющая локальный номер меньше максимального, то есть

$$m_k = a_k,\ k = s, s-1,\dots,j-1,\ m_j < a_j.$$

Если $m_s < a_s$, то вершина сама является опорной, если все $m_j = a_j$, то опорной назовем вершину верхнего слоя m_1.

Теперь можно сформулировать правило перехода к следующему элементу прямого произведения. Опорная вершина позволяет перейти к вершине со следующим глобальным номером, минуя переход к глобальным номерам. Из структуры графа видно, что для набора $(m_1,\dots,m_s)$ следующим будет набор $(m_1,\dots,m_j+1,1,\dots,1))$, здесь m_j — это опорная вершина.

Еще одна процедура, необходимая для реализации алгоритма, — восстановление локальных номеров по глобальному. Пусть m — глобальный номер $\left(1\le m\le\prod_{k=1}^{s}a_k\right)$, отметим, что номер вершины μ в слое k связан с локальным номером ρ тождеством

$$\mu=a_k\lambda+\rho,\ 1\le\rho\le a_k.$$

Опишем алгоритм вычисления $(m_1,\dots,m_s)$. На первом шаге:

$$m=a_s\lambda+\rho,\ 0\le\rho<a_s,$$

$$\text{если } \rho=0,\ \text{то } m_s=a_s,\ \lambda_s=\lambda-1,$$

$$\text{если } \rho>0,\ \text{то } m_s=\rho,\ \lambda_s=\lambda,$$

$$\dots$$

$$\text{на } k\text{-м шаге } \lambda_{k+1}=a_k\lambda+\rho,\ (k=s-1,s-2,\dots,1),$$

$$\text{если } \rho=0,\ \text{то } m_k=\rho_k,\ \lambda_k=\lambda-1,$$

$$\text{если } \rho_s>0,\ \text{то } m_s=\rho_k,\ \lambda_k=\lambda.$$

Эту процедуру легко обратить и восстановить глобальный номер по набору локальных номеров:

$$\lambda_1=m_1,\ \lambda_2=a_2(\lambda_1-1)+m_2,\ \dots,\ m=\lambda_s=a_s(\lambda_{s-1}-1)+m_s.$$

Вернемся к задаче нумерации наборов индексов. Так как $1\le r_k\le k$, то положим $G_1=(0)$, $G_2=(1)$, $G_3=(1,2),\dots$, $G_k=(1,2,\dots,k-1)$, $k=2,\dots,n$. Это позволит по номеру цепочки восстановить набор индексов и обратно.

3.5. Нумерация эквивалентных цепочек для фиксированной выборки

Чтобы занумеровать эквивалентные цепочки, необходимо перечислить все комбинации перестановок индексов в каждом пассивном интервале. Если имеется s пассивных интервалов длины q_j, $j=1,\dots,s$, то всего эквивалентных цепочек будет

$$Q=q_1!\times\dots\times q_s!.$$

Далее по номеру комбинации можно восстановить номера перестановок в каждом из интервалов. Для этого достаточно заметить, что комбинации перестановок индексов можно рассматривать как элементы прямого произведения множеств

$$G_j=(1,2,\dots,q_j!),\quad j=1,\dots,s.$$

В результате каждому номеру перемешивания t $(1\le t\le Q)$ будет сопоставлен набор $(t_1,\dots,t_s)$ номеров перестановок в каждом из пассивных интервалов. После этого остается перевести номер перестановки в саму перестановку.

Чтобы упростить обозначения, рассмотрим перестановки на стандартном множестве $(1,\dots,m)$. Заметим, что всякую перестановку можно представить $(w_1,\dots,w_m)$ как серию выборов:

— w_1 — на первом шаге одного элемента из m,
— w_2 — на втором шаге одного элемента из $m-1$,
— $\ldots$
— w_m — на m-м шаге берется последний оставшийся элемент.

Таким образом, выборку, отвечающую номеру m, можно рассматривать как последовательность локальных номеров элемента прямого произведения множеств

$$G_j = (1,2,\ldots,m+1-j), \quad j = 1,\ldots,m.$$

После этого остается перенести перестановку на множество индексов пассивного интервала: если $(1,\ldots,m) \to (s_1,\ldots,s_m)$, то $(r_{k_1},\ldots,r_{k_m}) \to (r_{k_{s_1}},\ldots,r_{k_{s_m}})$.

Таким образом, получено взаимно однозначное соответствие номеров перемешиваний и наборов перестановок в каждом пассивном интервале.

3.6. Модифицированный алгоритм. Нумерация эквивалентных цепочек для фиксированной выборки

Напомним структуру множества, образованного перемешиваниями в этом случае:

— пассивные интервалы $(r_{p_j+1},\ldots,r_{p_j+q_j}), \quad j = 1,2,\ldots,s,$
— стационарные отрезки в каждом из пассивных интервалов

$$(u_{j,0},u_{j,1}),(u_{j,1},u_{j,2}),\ldots,(u_{j,s-1},u_{j,s}), \quad u_{j,0}=1,\ u_{j,s}=q_j,$$

— существенные перестановки в стационарных отрезках.

Как было отмечено, общее число существенных перемешиваний равно

$$T = \prod_{j=1}^{s} \frac{q^{(j)}}{d_1^{(j)}!d_2^{(j)}!\ldots d_v^{(j)}!}.$$

На первом этапе надо рассмотреть прямое произведение множеств

$$G_j = \left(1,2,\ldots,\frac{q^{(j)}}{d_1^{(j)}!d_2^{(j)}!\ldots d_v^{(j)}!}\right), \quad j = 1,\ldots,s.$$

Это позволит номер существенного перемешивания t, $1 \le t \le T$, превратить в набор номеров существенных перестановок в каждом интервале $(t_1,\ldots,t_s)$.

Число существенных перестановок в пассивном интервале было вычислено ранее:

$$C_q^{d_1} \times C_{q-d_1}^{d_2} \times C_{q-d_1-d_2}^{d_3} \times C_{d_v}^{d_v} = \frac{q}{d_1!d_2!\ldots d_v!}$$

(ради упрощения обозначений индекс – номер пассивного интервала — здесь опущен). Правило нумерации позволяет получить из номера существенной перестановки набор номеров выборок для каждого стационарного отрезка. Для этого достаточно рассмотреть прямое произведение множеств

$$G_j = \left(1,2,\ldots,C_{q-u_{j-1}}^{u_j-u_{j-1}}\right), \quad j = 1,\ldots,s.$$

Далее необходимо по номеру выборки восстановить саму выборку. Эта процедура описана в [5].

Она основана на том, что всякое число m ($1 < m \le C_n^k$) единственным образом разлагается в сумму

$$m = C_{n_1}^k + C_{n_2}^{k-1} + \dots C_{n_j}^{k-j+1},$$

$$n > n_1 > \dots > n_j \ge n - j, \quad n_j = 1, \dots, k.$$

На основании разложения можно сформировать выборку (δ_i), $i = 1, 2, \dots, k$, здесь $\delta_i = 1$, если $i = n_j$, и $\delta_i = 0$ в остальных случаях. Неравенства, которым подчинены индексы n_i, позволяют взаимно однозначно дополнить эту выборку до выборок из n элементов по k. Геометрически это соответствует поднятию вершин единичного куба в $\mathbb{R}^n$, лежащих в плоскостях $x_{n-k} + x_{n-j+1} + \dots + x_{n-1} = j$, $j = 1, \dots, k$, в плоскость $x_{n-k} + x_{n-j+1} + \dots + x_{n-1} = k$, заменяя стоящие на крайних позициях нули на единицы.

В результате по номеру существенной перестановки будет сформирована серия выборок

$$(\varepsilon_1^{(j)}, \dots, \varepsilon_{q-u_{j-1}}^{(j)}),\ j = 1, 2, \dots, s, \quad \varepsilon_1^{(j)} + \dots + \varepsilon_{q-u_{j-1}}^{(j)} = u_j - u_{j-1}.$$

Эти выборки надо превратить в существенную перестановку на рассматриваемом пассивном интервале.

4. АЛГОРИТМ ВОССТАНОВЛЕНИЯ СУЩЕСТВЕННОЙ ПЕРЕСТАНОВКИ

1) Расставляем по местам, указанным в первой выборке, одинаковые индексы r_{u_1} из первого стационарного отрезка, на остальные позиции рассматриваемого пассивного интервала ставим нули (число нулей равно длине следующей выборки),

...

i) в пассивном интервале остается $q - u_{j-1}$ незаполненных позиций, в цикле (по w от 1 до q) по всем позициям пассивного интервала считаем число встретившихся нулей nmZ, если для очередного встретившегося номера $\varepsilon_{nmZ}^{(i)} = 1$, то на позицию w ставим индекс из стационарного интервала r_{u_i}, по завершении цикла в пассивном интервале останется $q - u_j$ незаполненных позиций, и можно переходить к следующему шагу,

...

s) на последнем шаге надо расставить индекс r_{u_s} по оставшимся незаполненными $u_s - u_{s-1}$ позициям.

После того как эта операция будет проведена на каждом пассивном интервале, получится набор индексов для очередной эквивалентной цепочки.

5. АЛГОРИТМ ПРЯМОГО ПЕРЕЧИСЛЕНИЯ

Накопленная информация о звездных цепочках позволяет иначе посмотреть на задачу поиска минимальной цепочки. Не нужно просматривать все цепочки, достаточно посмотреть по одному представителю в каждом классе цепочек, отличающихся только перемешиванием на пассивных интервалах.

Это можно сделать следующим образом (всюду далее предполагается, что длина цепочки фиксирована):

— цикл-1 — по количеству пассивных элементов в цепочке,

— цикл-2 — по всем возможным размещениям пассивных элементов.

Когда размещение фиксировано, нужно составить список пассивных интервалов. Затем надо рассмотреть элементы, не вошедшие в пассивные интервалы. Будем называть

их *активными*. У активного элемента индекс может быть любым (меньше номера самого элемента и не совпадающий с номером одного из пассивных элементов).

Далее надо перебрать все пассивные интервалы. Для каждого из них надо перечислить все возможные значения индексов так, чтобы один набор было невозможно перевести в другой перестановками в пассивных интервалах. Для этого достаточно выбрать индексы так, чтобы они образовывали неубывающую последовательность (тогда один набор невозможно перевести в другой перестановкой).

Все возможные комбинации этих выборов (прямое произведение) и дадут по одному представителю из каждого класса наборов индексов, отличающихся только перестановками в пассивных интервалах.

6. ОПИСАНИЕ ВАРИАНТОВ ВЫБОРА ИНДЕКСОВ В ПАССИВНОМ ИНТЕРВАЛЕ

Удобно провести это описание в терминах матрицы инцидентности. Пусть элементы пассивного интервала находятся на позициях от $p+1$ до $p+q$. Выбор индексов для этих элементов равносилен расстановке единиц в блоке матрицы, составленном из строк $p+1,\dots,p+q$ и столбцов $1,2,\dots,p$, при этом единицы нельзя ставить в столбцы, отвечающие пассивным элементам.

Пусть после удаления из блока столбцов пассивных элементов в нем останется t столбцов. Надо поставить единицы в каждой строчке так, чтобы единица в нижней строчке стояла под или левее единицы в верхней строке, то есть единицы в матрице образуют столбцы. Обозначим через v, $1 \le v \le \min(q,t)$, число таких столбцов и через $c_1,\dots,c_v$ номера столбцов матрицы, где находятся эти столбцы, а через $r_1,\dots,r_v$ номера строк, в которых находятся вершины этих столбцов, при этом

$$1 \le c_1 < c_2 < \dots < c_v \le t, \quad p+1 = r_1 < r_2 < \dots < r_v \le p+q.$$

Для перечисления всех допустимых перестановок надо организовать цикл перебора всех возможных значений параметра v. При фиксированном v всякая допустимая здесь расстановка единиц взаимно однозначно определяется числами $c_1,\dots,c_v$ и $r_1,\dots,r_v$. Наборы чисел в свою очередь можно интерпретировать как выборки v элементов из t и $v-1$ элементов из $q-1$. Общее число различных выборов расстановки единиц в матрице равно $C_t^v \times C_{q-1}^{v-1}$.

Занумеруем выборки индексов, отвечающие всем активным элементам и пассивным интервалам для всех значений v (сохраняя при этом списки соответствующих индексов). Прямое произведение эти объектов взаимно однозначно соответствует выборкам по одному представителю из каждого класса смежности.

Занумеруем стандартным способом элементы прямого произведения для всех значений числа пассивных элементов и для их всевозможных размещений. Проведем цикл по всем этим выборкам. На каждом шаге по глобальному номеру восстановим локальные номера, то есть индексы цепочек на соответствующих позициях. По индексам восстановим цепочку.

Всякая цепочка, имеющая ту же длину, эквивалентна одной из полученных цепочек.

7. ЗАКЛЮЧЕНИЕ

Настоящая работа является начальным этапом исследования. Использованное в работе достаточное условие эквивалентности цепочек довольно далеко отстоит от необходимого условия.

Для количественного описания этого различия удобно ввести определение. Пару цепочек назовем *существенно эквивалентными*, если одна переводится в другую перестановкой индексов в пассивных интервалах. Эквивалентные (имеющие одинаковый последний элемент) цепочки и существенно эквивалентные цепочки разбивают все множество цепочек на классы смежности. Классы смежности существенно эквивалентных цепочек являются подмножествами классов смежности эквивалентных цепочек. Изложенная выше конструкция позволяет выбрать по одному представителю из каждого класса смежности существенно эквивалентных цепочек.

Рассмотрим для примера цепочки длины 10. Всего таких цепочек 10! = 3628800. Классов смежности существенно эквивалентных цепочек 2249802.Классов смежности эквивалентных цепочек примерно 250. Для оценки ситуации важно, что распределение классов смежности существенно эквивалентных цепочек по классам смежности эквивалентных цепочек крайне неравномерно это видно из графиков (рис. 1), где по горизонтали отложено значение последнего элемента в цепочке, а по вертикали число классов смежности существенно эквивалентных цепочек, имеющих такой последний элемент.

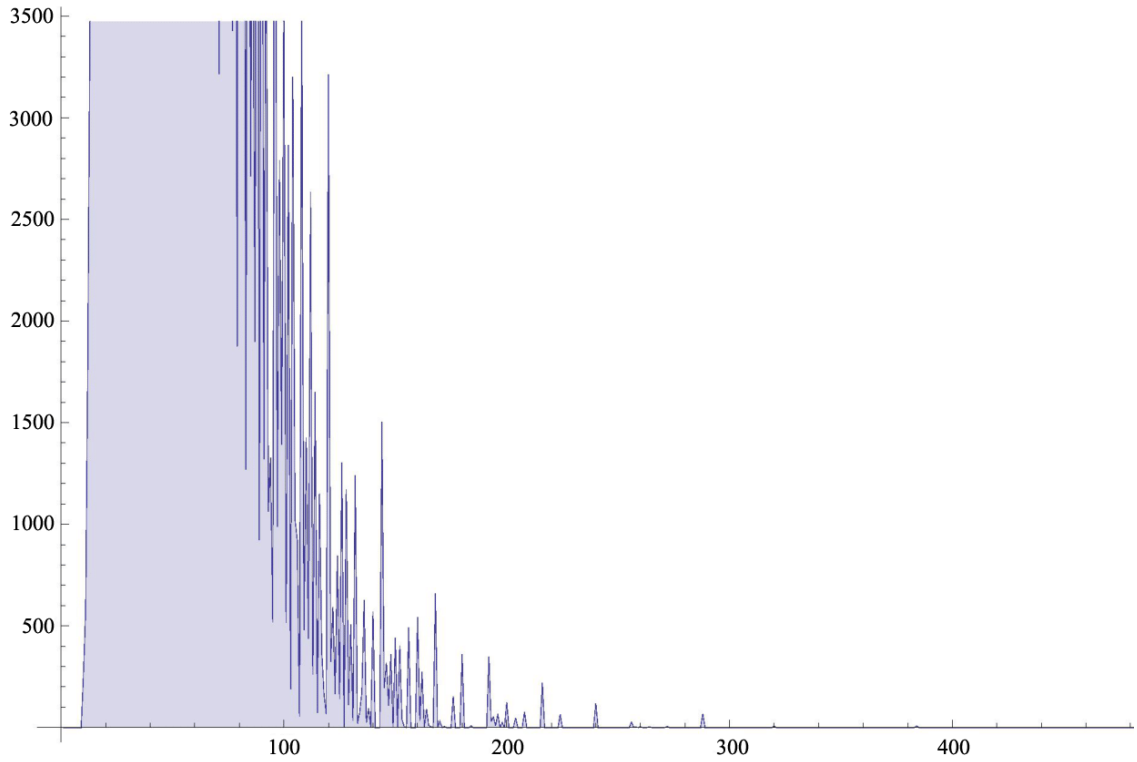

Рис. 1

Участок, где число классов особенно велико выделен отдельно (рис. 2). Поскольку в перспективе важны минимальные цепочки, и для чисел меньше 87 длины минимальных цепочек меньше 10, то классы смежности эквивалентных цепочек, содержащие много классов смежности существенно эквивалентных цепочек, находятся вне зоны, где могут находится минимальные цепочки длины 10. Изложенная здесь конструкция станет полезной, если удастся добавить к ней условие, исключающее из рассмотрения цепочки заведомо не являющиеся минимальными.

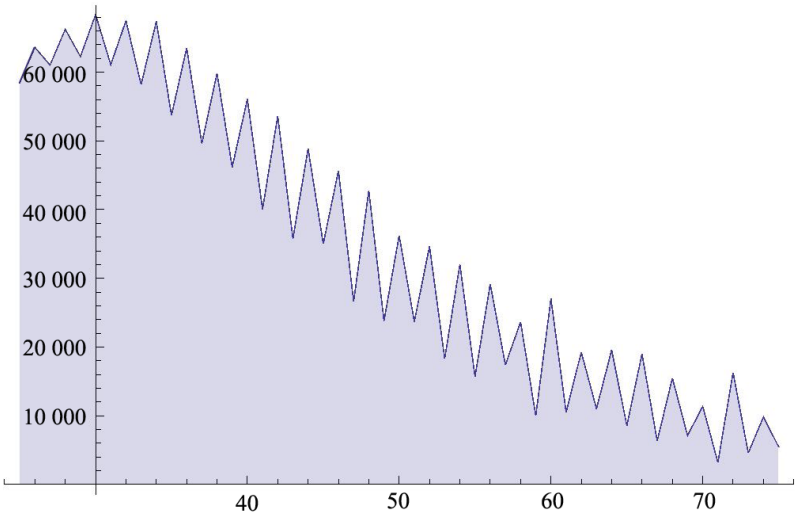

Рис. 2

Список литературы

1. *Кнут Д.* Искуство программирования. Том 1. М.: Мир, 1974.
2. *Pippenger N.* On the evaluation of powers and monomials // SIAM J. Comput. 1980. Vol. 9. № 2. P. 230–250. doi:10.1137/0209022
3. *Vassiliev N. N.* Evaluations of sparse polynomials and the synthesis of evaluating programs // Internat. Conf. on Comp. Algebra and Appl. Thtor. Phys., 1984. Dubna. P. 154–159.
4. *Кочергин В. В.* Улучшение оценки компьютерной сложности вычисления мономов и множеств степеней в задачах Беллмана и Кнута // Журнал прикладной и инженерной математики. 2015. Т. 9, № 1. С. 68–82.
5. *Коточигов А. М.* Алгоритм нумерации элементов выборки // Компьютерные инструменты в образовании. 2012. № 2. С. 34–39.

Поступила в редакцию 12.11.2019, окончательный вариант — 23.01.2020.

Коточигов Александр Михайлович, доктор физико-математических наук, профессор кафедры алгоритмической математики, ✉ amkotochigov@gmail.com

Сучков Андрей Игоревич, аспирант кафедры МО ЭВМ факультета компьютерных технологий и информатики СПбГЭТУ, suchkov3381@gmail.com

Computer tools in education, 2020
№ 1: 5–18
http://cte.eltech.ru
doi:10.32603/2071-2340-2020-1-5-18

A Method for Reducing Iteration in Algorithms for Building Minimal Additive Chains

Kotochigov A. M.[1], PhD, professor, ✉ amkotochigov@gmail.com
Suchkov A. I.[2], postgraduate student, suchkov3381@gmail.com

[1]Saint-Petersburg Electrotechnical University,
5, building 3, st. Professora Popova, 197376, Saint Petersburg, Russia

Abstract

Optimization of algorithms for computing the values of polynomials, more precisely, of monomials, is equivalent to the problem of constructing for a given number a minimal additive chain. To search for such chains, no algorithms are known except brute force. The increase in the complexity of the brute force algorithm is very large. Among chains of the same length there are a lot of equivalent ones, that is, those ending with the same number. The article provides a sufficient criterion for the equivalence of chains and shows how the use of the criterion reduces the procedure for the formation of all additive chains of a fixed length.

Keywords: *additive chain, equivalent chain, passive interval, essential randomisation.*

Citation: A. M. Kotochigov and A. I. Suchkov, "A Method for Reducing Iteration in Algorithms for Building Minimal Additive Chains," *Computer tools in education*, no. 1, pp. 5–18, 2020; doi:10.32603/2071-2340-2020-1-5-18 (in Russian).

References

1. D. Knuth, *Iskusstvo programmirovaniya* [The Art of Computer Programming], K. I. Babenko and Yu. M. Bayakovskii, eds., vol. 1, Moscow: Iz-vo Mir, 1974 (in Russian).
2. N. Pippenger, "On the evaluation of powers and monomials," *SIAM J. Comput.*, vol. 9, no. 2, pp. 230–250, 1980; doi:10.1137/0209022
3. N. N. Vassiliev, "Evaluations of sparse polynomials and the synthesis of evaluating programs," in *Proc. Internat. Conf. on Comp. Algebra and Appl. Thtor. Phys.*, Dubna, 1984, pp. 154–159.
4. V. V. Kochergin, "Utochnenie otsenok slozhnosti vychisleniya odnochlenov i naborov stepenei v zadachakh Bellmana i Knuta" [Improvement of the estimates of the computational complexity for monomials and sets of powers in Bellman's and Knuth's problems], *J. Appl. Industr. Math.*, vol. 9, no. 1, pp. 68–82, 2015 (in Russian); doi: 10.1134/S1990478915010081
5. A. M. Kotochigov, "Algoritmy fraktal'nogo analiza izobrazhenii" [Fractal Image Analysis Algorithms], *Computer tools in education*, no. 2, pp. 34–39, 2012 (in Russian).

Received 12.11.2019, the final version — 23.01.2020.

Aleksandr M. Kotochigov, PhD, professor of Department of Algorithmic Mathematics, ✉ amkotochigov@gmail.com

Andrey I. Suchkov, postgraduate student of the Department of computer science of the Faculty of Computer Science and Technology of ETU "LETI", suchkov3381@gmail.com