Фрактран: простой универсальный язык программирования для арифметики

Дж. Конвей

§ 1. Бесплатные образцы Фрактрана для вас

Игра во *Фрактран* состоит в следующем. Дан список дробей

$$f_1,\ f_2,\ \ldots,\ f_k$$

и начальное целое число N. Вы последовательно умножаете текущее целое число (начиная с N) на первую дробь f_i из списка, для которой результат будет целым. Если такой дроби нет, игра *окончена*.

(Формально последовательность $\{N_n\}$ строится по следующему правилу: $N_0 = N$, $N_{n+1} = f_i N_n$, где i $(1 \leqslant i \leqslant k)$ — наименьшее i, для которого $f_i N_n$ целое, если такое i существует.)

Пример 1 (PRIMEGAME). Пусть список дробей имеет вид

$$\frac{17}{91}\ \frac{78}{85}\ \frac{19}{51}\ \frac{23}{38}\ \frac{29}{33}\ \frac{77}{29}\ \frac{95}{23}\ \frac{77}{19}\ \frac{1}{17}\ \frac{11}{13}\ \frac{13}{11}\ \frac{15}{2}\ \frac{1}{7}\ \frac{55}{1}$$

и игра начинается при $N = 2$. Тогда появятся в точности те степени двойки, показатели которых — простые числа, причём в порядке их возрастания: $2^2, 2^3, 2^5, 2^7, 2^{11}, 2^{13}, 2^{17}, 2^{19}, 2^{23}, 2^{29}, \ldots$

Пример 2 (PIGAME). Пусть список дробей имеет вид

$$\frac{365}{46}\ \frac{29}{161}\ \frac{79}{575}\ \frac{679}{451}\ \frac{3159}{413}\ \frac{83}{407}\ \frac{473}{371}\ \frac{638}{355}\ \frac{434}{335}\ \frac{89}{235}\ \frac{17}{209}\ \frac{79}{122}$$

$$\frac{31}{183}\ \frac{41}{115}\ \frac{517}{89}\ \frac{111}{83}\ \frac{305}{79}\ \frac{23}{73}\ \frac{73}{71}\ \frac{61}{67}\ \frac{37}{61}\ \frac{19}{59}\ \frac{89}{57}\ \frac{41}{53}\ \frac{833}{47}\ \frac{53}{43}$$

$$\frac{86}{41}\ \frac{13}{38}\ \frac{23}{37}\ \frac{67}{31}\ \frac{71}{29}\ \frac{83}{19}\ \frac{475}{17}\ \frac{59}{13}\ \frac{41}{291}\ \frac{1}{7}\ \frac{1}{11}\ \frac{1}{1024}\ \frac{1}{97}\ \frac{89}{1}$$

Conway J. H. FRACTRAN — a simple universal programming language for arithmetic // Open Problems Commun. Comput. 1986. P. 4–26.

Составители благодарны Паулю Зиверту за тщательную работу со статьёй.

и игра начинается при $N = 2^n$. Тогда следующая степень двойки в последовательности будет равна $2^{\pi(n)}$:

$$\begin{array}{l} n = 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20\ \ldots, \\ \pi(n) = 3\ 1\ 4\ 1\ 5\ 9\ 2\ 6\ 5\ 3\ \ 5\ \ 8\ \ 9\ \ 7\ \ 9\ \ 3\ \ 2\ \ 3\ \ 8\ \ 4\ \ 6\ \ \ldots \end{array}$$

Для произвольного натурального числа n значение $\pi(n)$ равно n-му знаку в десятичной записи дробной части числа π.

Пример 3 (POLYGAME). Пусть список дробей имеет вид

$$\frac{583}{559}\ \frac{629}{551}\ \frac{437}{527}\ \frac{82}{517}\ \frac{615}{329}\ \frac{371}{129}\ \frac{1}{115}\ \frac{53}{86}\ \frac{43}{53}\ \frac{23}{47}$$

$$\frac{341}{46}\ \frac{41}{43}\ \frac{47}{41}\ \frac{29}{37}\ \frac{37}{31}\ \frac{299}{29}\ \frac{47}{23}\ \frac{161}{15}\ \frac{527}{19}\ \frac{159}{7}\ \frac{1}{17}\ \frac{1}{13}\ \frac{1}{3}$$

и игра начинается при $N = c2^{2^n}$. Если игра заканчивается на 2^{2^m}, то положим $f_c(n) = m$. В противном случае оставим $f_c(n)$ неопределённым. Тогда среди $f_0, f_1, f_2, \ldots$ появляется любая вычислимая функция.

§ 2. Каталог

Отметим, что для некоторых весьма интересных функций «списочные числа» c легко вычисляются. В таблице 1 и пояснениях к ней приведены f_c для всех c, у которых наибольший нечётный делитель меньше, чем $2^{10} = 1024$.

Кроме того,

$$\begin{aligned} &f_{2^kA} = f_0; \\ &f_{2^kB} = f_{2^k}; \quad f_{2^kB'} = f_{2^{k+1}}; \\ &f_{2^kC} = f_{77}; \quad f_{2^kC'} = f_{847}; \\ &f_{2^kD} = f_{133}\ (k = 0) \quad \text{или} \quad f_0\ (k > 0); \\ &f_{2^kE} = f_{255}\ (k = 0) \quad \text{или} \quad f_{2^k}\ (k > 0). \end{aligned}$$

Здесь

A любое нечётное число, меньшее чем 1024 и не приведённое ниже;

B 1, 3, 9, 13, 17, 27, 39, 45, 51, 81, 105, 115, 117, 135, 145, 153, 155, 161, 169, 185, 195, 203, 205, 217, 221, 235, 243, 259, 287, 289, 315, 329, 345, 351, 405, 435, 459, 465, 483, 507, 555, 585, 609, 615, 651, 663, 705, 729, 777, 861, 945, 975, 987, 1017, … ;

B' 165, 495, … ;

Таблица 1. Списочные числа
(здесь n — произвольное неотрицательное целое число)

c	Все определённые значения f_c
0	—
1	$n \to n$
2	$0 \to 1$
4	$0 \to 2$
8	$1 \to 2$
16	$2 \to 3$
64	$1 \to 3$
77	$n \to 0$
128	$0 \to 3$
133	$0 \to 0$
255	$n+1 \to n+1$
256	$3 \to 4$
847	$n \to 1$
37 485	$0 \to 0,\ n+1 \to n$
2 268 945	$n \to n+1$
2^k	$a \to b$ при $2^b - 2^a = k$
$7 \cdot 11^{2^k}$	$n \to k$
$\frac{15}{7} \cdot 1029^{2^{k-1}}$	$n \to n+k$
c_π	$n \to \pi(n)$

C $77, 91, 231, 273, 385, 455, 539, 1015, \ldots$;

C' $847, 1001, \ldots$;

D $133, 285, 399, 665, 855, \ldots$;

E $255, \ldots$

На рис. 1 приведена формула для значения c, при котором $f_c(n)$ совпадает с описанной выше функцией $\pi(n)$.

§ 3. Избегайте случайного выбора

В теории эффективности вычислений многие работы написаны авторами, чьи интересы относятся больше к логике, чем к теории вычислений, и потому собственно вычислительные аспекты теории в них редко излагаются изящно. Вероятно, любая эффективная ну-

$$2^{100!}+2^{\frac{365}{46}\cdot 101\cdot 100!}+2^{\frac{29}{161}\cdot 101^2\cdot 100!}+2^{\frac{79}{575}\cdot 101^3\cdot 100!}+2^{\frac{7}{451}\cdot 101^4\cdot 100!}+$$
$$+2^{\frac{3159}{413}\cdot 101^5\cdot 100!}+2^{\frac{83}{407}\cdot 101^6\cdot 100!}+2^{\frac{473}{371}\cdot 101^7\cdot 100!}+2^{\frac{638}{355}\cdot 101^8\cdot 100!}+2^{\frac{434}{335}\cdot 101^9\cdot 100!}+$$
$$+2^{\frac{89}{235}\cdot 101^{10}\cdot 100!}+2^{\frac{17}{209}\cdot 101^{11}\cdot 100!}+2^{\frac{79}{122}\cdot 101^{12}\cdot 100!}+2^{\frac{31}{183}\cdot 101^{13}\cdot 100!}+2^{\frac{41}{115}\cdot 101^{14}\cdot 100!}+$$
$$+2^{\frac{517}{89}\cdot 101^{15}\cdot 100!}+2^{\frac{111}{83}\cdot 101^{16}\cdot 100!}+2^{\frac{305}{79}\cdot 101^{17}\cdot 100!}+2^{\frac{23}{73}\cdot 101^{18}\cdot 100!}+2^{\frac{73}{71}\cdot 101^{19}\cdot 100!}+$$
$$+2^{\frac{61}{67}\cdot 101^{20}\cdot 100!}+2^{\frac{37}{61}\cdot 101^{21}\cdot 100!}+2^{\frac{19}{59}\cdot 101^{22}\cdot 100!}+2^{\frac{89}{57}\cdot 101^{23}\cdot 100!}+2^{\frac{41}{53}\cdot 101^{24}\cdot 100!}+$$
$$+2^{\frac{833}{47}\cdot 101^{25}\cdot 100!}+2^{\frac{53}{43}\cdot 101^{26}\cdot 100!}+2^{\frac{86}{41}\cdot 101^{27}\cdot 100!}+2^{\frac{13}{38}\cdot 101^{28}\cdot 100!}+2^{\frac{23}{37}\cdot 101^{29}\cdot 100!}+$$
$$+2^{\frac{67}{31}\cdot 101^{30}\cdot 100!}+2^{\frac{71}{29}\cdot 101^{31}\cdot 100!}+2^{\frac{83}{19}\cdot 101^{32}\cdot 100!}+2^{\frac{475}{17}\cdot 101^{33}\cdot 100!}+2^{\frac{59}{13}\cdot 101^{34}\cdot 100!}+$$
$$+2^{\frac{41}{3}\cdot 101^{35}\cdot 100!}+2^{\frac{1}{7}\cdot 101^{36}\cdot 100!}+2^{\frac{1}{11}\cdot 101^{37}\cdot 100!}+2^{\frac{1}{1024}\cdot 101^{38}\cdot 100!}+2^{101^{39}\cdot 100!}$$

$$3\times 5^{2^{89\cdot 101!}+2^{90\cdot 101!}}\times 17^{101!-1}\times 23$$

Рис. 1. Константа c_π

мерация вычислимых функций достаточно сложна, чтобы заполнить главу в книге, и мы могли читать, что «явное вычисление порядкового номера любой интересной функции, разумеется, невыполнимо на практике». Многие подобные недостатки происходят от плохого выбора используемой модели вычислений.

На наш взгляд, как раз потому, что конкретная вычислительная модель не представляет большого интереса с точки зрения логики, она должна быть выбрана тщательно. Логические аспекты станут лишь яснее, если читателю не придётся их вычленять из корявой программы, написанной на ужасном языке. И тогда мы сможем «продать» нашу теорию более широкой аудитории, представляя ей простые и впечатляющие примеры. (Именно в связи с этим мы используем лёгкий для понимания термин «вычислимая функция» как синоним обычного «частично рекурсивная функция».)

§ 4. Только Фрактран обладает этими звёздными качествами

Фрактран — это простой язык теоретического программирования для арифметики, лишённый вышеописанных недостатков.

- *Делает рабочий день поистине лёгким*!

Фрактран не нуждается в сложном справочнике программиста — весь его синтаксис выучивается за 10 секунд, и практически сразу можно писать программы для достаточно сложных и интересных функций.

- *Получает эти функции в чистом виде*!
  В каждый момент Фрактран работает с единственным целым числом — неопытному программисту не нужно разбираться ни с какими запутанными «лентами» и прочими чуждыми объектами.
- *Пригоден для любого компьютера в продаже*!
  Ваши старые машины (машина Тьюринга и т. д.) легко допускают симуляцию любых Фрактран-программ, и обычно бывает даже легче симулировать другие машины посредством Фрактран-программы.
- *Удивительно простая универсальная программа*!
  Создав Фрактран-программу, которая симулирует любую другую Фрактран-программу, мы получили простую универсальную Фрактран-программу, описанную в теореме 3.

§ 5. Ваша гарантия от PRIMEGAME!

Отчасти жалко, когда теряют таинственность наши программы — например, PRIMEGAME. Однако в работе [2] хорошо сказано[1]: «Математик — это фокусник, который раздаривает свои секреты», так что мы сейчас докажем теорему 1.

Чтобы пояснить рис. 2, обозначим наши дроби так:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
$\frac{17}{91}$	$\frac{78}{85}$	$\frac{19}{51}$	$\frac{23}{38}$	$\frac{29}{33}$	$\frac{77}{29}$	$\frac{95}{23}$	$\frac{77}{19}$	$\frac{1}{17}$	$\frac{11}{13}$	$\frac{13}{11}$	$\frac{15}{2}$	$\frac{1}{7}$	$\frac{55}{1}$

и заметим, что

$$AB = \frac{2\cdot 3}{5\cdot 7}, \quad EF = \frac{7}{3}, \quad DG = \frac{5}{2}.$$

Пусть n и d — натуральные числа, $0 < d < n$, и пусть $n = qd + r$ $(0 \leqslant r < d)$. На рис. 2 показано, как действует PRIMEGAME на число $5^n 7^d 13$. Мы видим, что результат равен $5^n 7^{d-1} 13$ или $5^{n+1} 7^n 13$, если d соответственно делит или не делит n. При этом единственный случай, когда появляется степень двойки, — число $2^n 7^{d-1}$ при $d = 1$.

[1] Ссылка фразы на саму себя. Автор успешно развивает эту идею в статье «Каверзная задача», см. с. 315–324 наст. издания. — *Прим. составителей.*

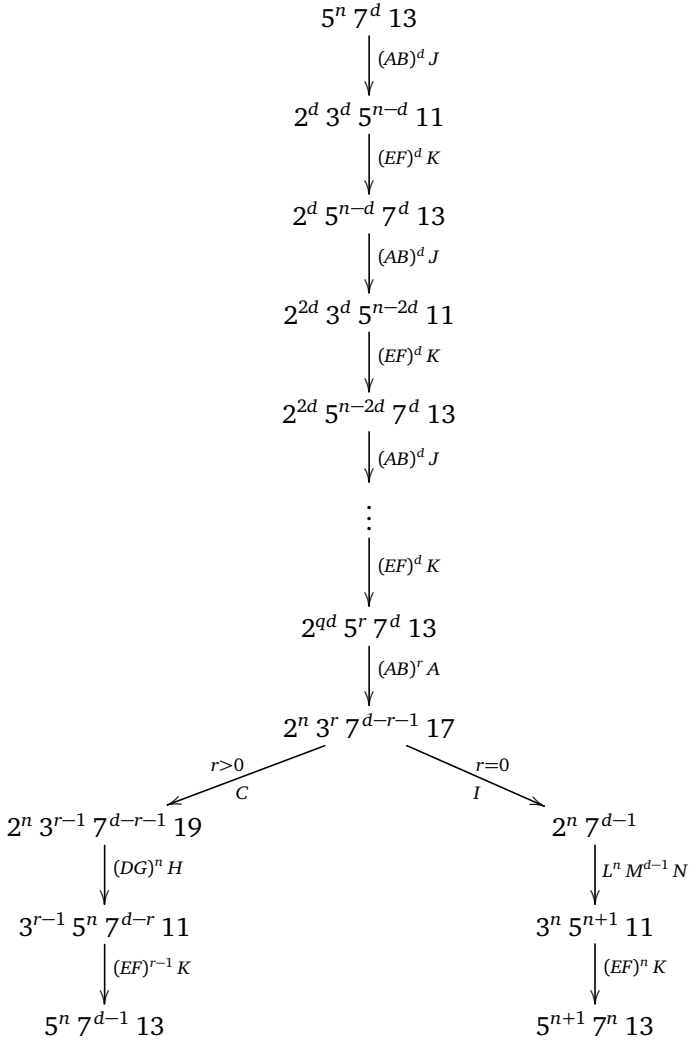


Рис. 2. Действие программы PRIMEGAME

Таким образом, если игра начинается с $5^n 7^{n-1} 13$, то она проверяет все натуральные числа, начиная с n, по направлению к 1, пока не найдёт делитель числа n, а после этого продолжается с $n+1$ вместо n. Степень двойки 2^n появляется, лишь если наибольший собственный делитель числа n равен 1, то есть если n простое.

§ 6. Фрактран: бесплатное предварительное предложение

Фрактран-программа может состоять из любого количества строк (команд), и типичный вид строки таков:

$$\text{строка 13:}\quad \frac{2}{3}\to 7,\quad \frac{4}{5}\to 14.$$

В этом случае компьютер заменяет текущее значение числа N на $\frac{2}{3}N$, если это число также целое, и переходит на строку 7. Если $\frac{2}{3}N$ не целое, но $\frac{4}{5}N$ целое, то надо заменить N на $\frac{4}{5}N$ и перейти на строку 14. Если же ни $\frac{2}{3}N$, ни $\frac{4}{5}N$ не целые, то на строке 13 мы *останавливаемся*.

Более общим образом, строка Фрактран-программы имеет вид

$$\text{строка } n\text{:}\quad \frac{p_1}{q_1}\to n_1,\quad \frac{p_2}{q_2}\to n_2,\ \ldots,\quad \frac{p_k}{q_k}\to n_k.$$

В этом случае компьютер заменяет N на $\frac{p_i}{q_i}N$, где i $(1\leqslant i\leqslant k)$ — наименьшее, для которого новое значение целое. Затем компьютер переходит к строке n_i; если же все $\frac{p_i}{q_i}N$ дробные, то на строке n программа *останавливается*. (Строка с $k=0$ запрещена и означает безусловную остановку.)

Фрактран-программа, имеющая ровно n строк, называется Фрактран-n-программой. Введём соглашение, что $\frac{1}{2}$-строка — это строка, переход на которую невозможен. (Разумные программы могут содержать не более одной $\frac{1}{2}$-строки, а именно начальную.)

Запись

$$\left[\frac{p_1}{q_1}\ \frac{p_2}{q_2}\ \ldots\ \frac{p_k}{q_k}\right]$$

означает Фрактран-1-программу

$$\text{строка 1:}\quad \frac{p_1}{q_1}\to 1,\quad \frac{p_2}{q_2}\to 1,\ \ldots,\quad \frac{p_k}{q_k}\to 1.$$

Мы увидим, что любая Фрактран-программа симулируется Фрактран-1-программой, для которой начальное число — подходящее кратное исходного начального числа. Это кратное можно сделать равным 1, если использовать Фрактран-$1\frac{1}{2}$-программу. Это программа

$$\text{строка 0:}\quad \frac{P_1}{Q_1}\to 1,\quad \frac{P_2}{Q_2}\to 1,\ \ldots,\quad \frac{P_j}{Q_j}\to 1,$$

$$\text{строка 1:}\quad \frac{p_1}{q_1}\to 1,\quad \frac{p_2}{q_2}\to 1,\ \ldots,\quad \frac{p_k}{q_k}\to 1.$$

Будем обозначать её

$$\frac{P_1}{Q_1}\ \frac{P_2}{Q_2}\ \cdots\ \frac{P_j}{Q_j}\ \left[\frac{p_1}{q_1}\ \frac{p_2}{q_2}\ \cdots\ \frac{p_k}{q_k}\right].$$

Отметим, что Фрактран-$1\frac{1}{2}$-программа

$$m[f_1, f_2, \ldots, f_k],$$

стартующая в N, симулирует Фрактран-1-программу

$$[f_1, f_2, \ldots, f_k],$$

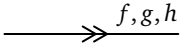
стартующую в mN.

Будем обычно предполагать по умолчанию, что наши Фрактран-программы применяются лишь к числам N, простые делители которых содержатся среди делителей числителей и знаменателей используемых дробей.

§ 7. Руководство для начинающего Фрактран-программиста

Целесообразно писать Фрактран-программы в виде блок-схем, где узлы обозначают строки, а стрелки между ними помечаются соответствующими дробями. Наконечники этих стрелок будут различаться:

соответственно убывающему приоритету данного узла, и если у строки несколько опций с дробями f, g, h имеют последовательный приоритет, то мы часто будем объединять их в одну стрелку:



Различные простые числа, появляющиеся в числителях и знаменателях дробей, могут рассматриваться как регистры хранения. Если текущее целое число имеет вид

$$N = 2^a 3^b 5^c 7^d \ldots,$$

то мы говорим, что

регистр 2 хранит a, или $r_2 = a$,
регистр 3 хранит b, или $r_3 = b$,
регистр 5 хранит c, или $r_5 = c$,
регистр 7 хранит d, или $r_7 = d$,
и т. д.

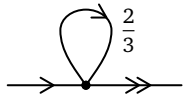
Строки Фрактран-программы могут также рассматриваться как команды, меняющие содержимое этих регистров на небольшие величины. Главное ограничение: никакой регистр не может содержать отрицательное число. Таким образом, строка вида

$$\text{строка 13:}\quad \frac{2}{3} \to 7,\quad \frac{4}{5} \to 14$$

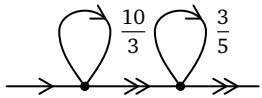
либо *заменяет* r_2 *на* r_2+1, r_3 *на* r_3-1 (при $r_3>0$),
либо *заменяет* r_2 *на* r_2+2, r_5 *на* r_5-1 (при $r_5>0$),
либо *останавливает программу* (при $r_3=r_5=0$).

На наших схемах используются непомеченные стрелки, когда соответствующие функции равны 1. Крошечная стрелка, входящая в узел, показывает, что этот узел будет начальным, а выходящая из узла — что в этом узле может произойти остановка. Несколько простых примеров должны убедить читателя, что Фрактран действительно обладает универсальными вычислительными возможностями. (Читатели, знакомые с понятием регистровой машины Минского, увидят, что Фрактран симулирует её без труда.)

Программа

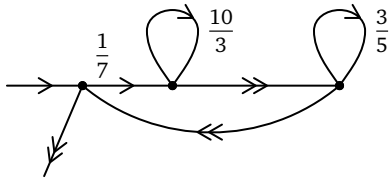


— деструктивный сумматор: начав с $r_2=a$, $r_3=b$, она останавливается при $r_2=a+b$, $r_3=0$. Можно сделать его менее деструктивным, использовав регистр 5 как рабочую ячейку: если программа



стартует с $r_2=a$, $r_3=b$, $r_5=0$, то она останавливается при $r_2=a+b$, $r_3=b$, $r_5=0$.

Умножение можно реализовать как многократное сложение: если программа



стартует с $r_2 = a$, $r_3 = b$, $r_5 = 0$, $r_7 = c$, то она останавливается при $r_2 = a + bc$, $r_3 = b$, $r_5 = r_7 = 0$. Добавим в исходном/финальном узле $1/3$ («чистку от троек») и сформулируем результат по правилам Фрактрана:

$$\begin{aligned}
&\text{строка 1:} \quad \frac{1}{7} \to 2, \quad \frac{1}{3} \to 1,\\
&\text{строка 2:} \quad \frac{10}{3} \to 2, \quad \frac{1}{1} \to 3,\\
&\text{строка 3:} \quad \frac{3}{5} \to 3, \quad \frac{1}{1} \to 1.
\end{aligned}$$

Начав со строки 1 при $N = 3^b 7^c$, программа останавливается на строке 1 при $N = 2^{bc}$.

Если к этой программе добавлена впереди новая

$$\text{строка 0:} \quad \frac{21}{2} \to 0, \quad \frac{1}{1} \to 1,$$

то, начав со строки 0 при $N = 2^n$, программа останавливается на строке 1 при $N = 2^{n^2}$.

§ 8. Как симулировать Фрактран на Фрактране-1

С помощью Фрактрана-1 можно симулировать любые программы на Фрактране. Прежде всего нужно очистить программу от петель — способ мы поясним позже, — а затем пометить её строки (узлы) простыми числами $P, Q, R, \ldots$, превосходящими любой простой делитель числителей и знаменателей её дробей. Строка

$$P\colon \quad \frac{a}{b} \to Q, \quad \frac{c}{d} \to R, \quad \frac{e}{f} \to S, \quad \ldots$$

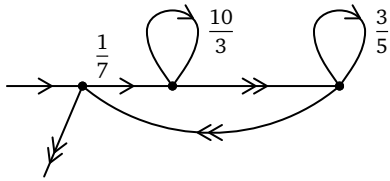
симулируется во Фрактран-1-программе дробями

$$\frac{aQ}{bP} \quad \frac{cR}{dP} \quad \frac{eS}{fP} \quad \cdots$$

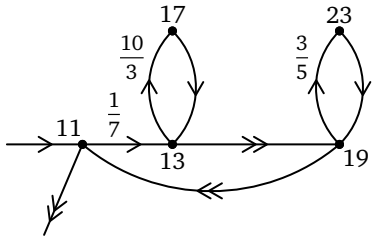
в указанном порядке. Если Фрактран-0-программа стартует с числа N в состоянии P и останавливается в строке Q на числе M, то симулирующая Фрактран-1-программа стартует с PN и останавливается на QM.

Замечание производителя. Наша гарантия недействительна, если вы применяете Фрактран-1 описанным способом для симуляции Фрактран-программы, имеющей петли в нескольких узлах. Такие петли можно устранить, расщепляя один узел на два.

Третий из наших примеров:



принимает вид



при расщеплении каждого из двух узлов этим способом. Новые узлы помечены простыми числами 11, 13, 17, 23. Соответственно, эта программа симулируется Фрактран-1-программой

$$\left[\frac{13}{77}\ \ \frac{170}{39}\ \ \frac{19}{13}\ \ \frac{13}{17}\ \ \frac{69}{95}\ \ \frac{11}{19}\ \ \frac{19}{23}\right].$$

Если начать с $N = 2^a 3^b 7^c 11$, то программа остановится при $N = 2^{a+bc} 3^b 11$. (Множители 11 здесь соответствуют начальному и финальному состояниям симулируемой программы.)

Заметим, что разрешается пометить одно из состояний числом 1 вместо большого простого. Дроби, отвечающие переходам из этого состояния, должны стоять (в том же порядке) в *конце* Фрактран-1-программы. В этом случае петли допускаются в узле 1 — при условии, что их приоритет ниже, чем у любого другого перехода. Таким образом, Фрактран-1-программа

$$\left[\frac{170}{39}\ \ \frac{19}{13}\ \ \frac{13}{17}\ \ \frac{69}{95}\ \ \frac{1}{19}\ \ \frac{19}{23}\ \ \frac{13}{7}\ \ \frac{1}{3}\right]$$

симулирует предыдущую программу с петлёй 1/3, присоединённой к начальному/финальному узлу, который теперь помечен числом 1. Эта программа, начав с $3^b 7^c$, останавливается в 2^{bc}.

Всегда можно очистить заданную Фрактран-программу от петель и обеспечить, чтобы 1 было её единственным финальным узлом. Следовательно, можно симулировать её Фрактран-1-программой, которая стартует с PN и останавливается в M, если исходная программа

стартует с N и останавливается в M. Как мы отметили в § 6, также возможна симуляция Фрактран-$1\frac{1}{2}$-программой

$$P[\ldots],$$

которая стартует с N и останавливается в M.

§ 9. Ваша гарантия от PIGAME

Теперь докажем теорему 2, равносильную следующему утверждению: если программа

$$\left[\frac{365}{46}\ \ \frac{29}{161}\ \ \cdots\ \ \frac{1}{11}\ \ \frac{1}{1024}\right]$$

(полученная из PIGAME устранением множителей, равных 97, а также последней дроби 89/1) стартует с $2^n \cdot 89$, то она останавливается в $2^{\pi(n)}$. Эта Фрактран-1-программа получена из Фрактран-программы на рис. 3 методом, намеченным в предыдущем параграфе. Пары узлов 13 & 59, 29 & 71, 23 & 73, 31 & 67 и 43 & 53 вначале содержали все петли.

Дадим лишь набросок работы этой программы, разделив её на три фазы. Первая фаза заканчивается, когда программа впервые достигает узла 37, вторая — при первом приходе в узел 41, а третья — когда программа останавливается в узле 1.

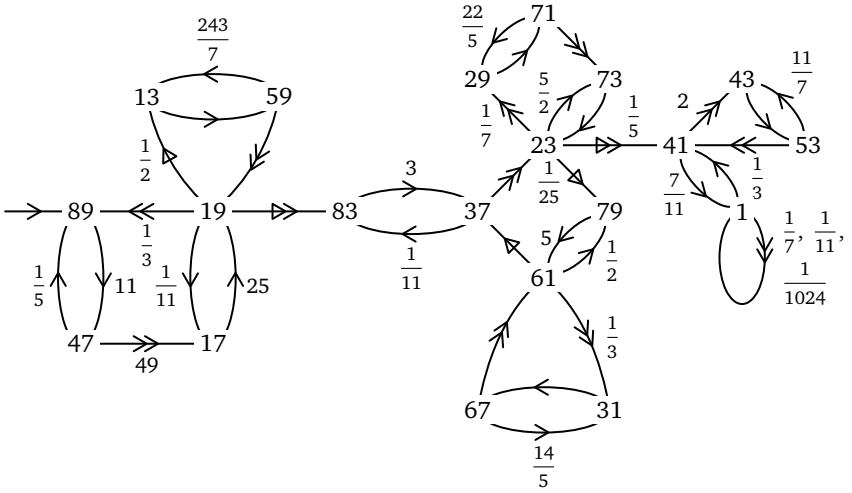


Рис. 3. Фрактран-программа для вычисления знаков числа π

Первая фаза, которая начинается в узле 89 при содержимом регистров

$$r_2 = n, \quad r_3 = r_5 = r_7 = r_{11} = 0,$$

заканчивается в узле 37 при содержимом регистров

$$r_2 = 0, \quad r_3 = 1, \quad r_5 = E, \quad r_7 = 2 \cdot 10^n, \quad r_{11} = 0,$$

где E — очень большое чётное число. Чтобы убедиться в этом, на минуту забудем про регистры 5 и 11 и заметим, что вначале мы получаем $r_7 = 2$. В дальнейшем каждый обход вокруг треугольной области умножает r_7 на 5 и помещает его в r_3, а затем r_3 при обходах вокруг квадрата удваивается и возвращается в r_7. Это делается n раз, так что в конце этой фазы мы имеем $r_7 = 2 \cdot 10^n$, что и требовалось.

После первого обхода вокруг квадрата регистр r_5 содержит 4, а каждый последующий обход по меньшей мере удваивает число в этом регистре, сохраняя его чётным. На последнем этапе мы обходим эту область 10^n раз и получаем в r_5 чётное число $E \geqslant 4 \cdot 2^{10^n}$. Легко проверить, что регистры 2, 3 и 11 в конце содержат указанные значения.

После второй фазы мы должны получить

$$r_2 = r_5 = r_7 = 0,$$
$$r_3 = 2 \cdot 10^n \cdot E(E-2)(E-2)(E-4)(E-4)(E_6) \ldots 4 \cdot 4 \cdot 2 \cdot 2 \triangleq N,$$
$$r_{11} = 1 \cdot (E-1)(E-1)(E-3)(E-3)(E-5)(E-5) \ldots 5 \cdot 3 \cdot 3 \cdot 1 \triangleq D.$$

Это проверяется без труда; важнейший пункт состоит в том, что при каждом пребывании в верхней области r_7 умножается на r_5 и сохраняется в r_{11} (причём r_5 не меняется, а r_7 обращается в нуль), тогда как в нижней области мы аналогично умножаем r_3 на r_5 и сохраняем в r_7, а затем (в левой части) переносим r_{11} назад в r_3. Регистр 5 при переходе из верхней области в нижнюю уменьшается на 1, но при $r_5 = 1$ мы вместо этого очищаем его и идём в узел 41, переходя в третью фазу.

Напомним, что *формула Валлиса* имеет вид

$$\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \cdot \frac{8}{9} \cdot \frac{10}{9} \cdot \frac{10}{11} \cdot \ldots,$$

где каждая следующая дробь получается увеличением на 1 поочерёдно числителя и знаменателя. Если мы ограничимся дробями, числитель и знаменатель которых не превышают K, то получим приближение π_K к π с точностью не меньше π/K. Отсюда следует, что $N/D = 10^n \pi_E$, где π_E — очень хорошее приближение к π. Настолько хорошее, что n-я десятичная цифра в дробной части π_E такая же, как

в π. Эту цифру можно получить как остаток от деления N/D на 10, и легко проверить, что третья фаза нашей программы делает именно это, ставит ответ в регистр 2 и очищает остальные регистры.

Утверждение о точности n-го десятичного знака в π_E нетривиально. При $n = 0$ наше приближение π_E равно $\pi_4 = 32/9$. При $n = 1, 2$ получаем $|\pi_E - \pi| < \pi/(4 \cdot 2^{10})$, что меньше чем $1/1000$, так как $\pi = 3{,}141\ldots$, поэтому первая и вторая цифры в дробной части числа π_E будут правильными.

При $n \geqslant 3$ погрешность в π_E не превышает

$$\frac{\pi}{4 \cdot 2^{10^n}} < \frac{1}{(1000)^{10^{n-1}}} = 10^{-3 \cdot 10^{n-1}} < 10^{-42n}.$$

Нужный результат теперь следует из известной оценки иррациональности числа π (Малер [4]): если p/q — дробное число в несократимом виде, то

$$\left|\pi - \frac{p}{q}\right| > \frac{1}{q^{42}}.$$

§ 10. Как применять нашу универсальную программу

В этом параграфе мы докажем теорему 3 с помощью остроумной леммы, принадлежащей Джону Рикарду (John Rickard). Будем называть Фрактран-1-программу $[f_1, f_2, \ldots, f_k]$ *монотонной*, если $f_1 < f_2 < f_3 < \ldots < f_k$.

Лемма. *Любую Фрактран-1-программу можно симулировать монотонной программой, которая стартует и останавливается на тех же числах.*

Доказательство. Выберем новое простое число P, которое больше, чем отношение любых двух f_i, а также больше, чем обратное к любому f_i. Тогда программа

$$\left[\frac{1}{P}, Pf_1, P^2f_2, P^3f_3, \ldots, P^kf_k\right]$$

симулирует $[f_1, f_2, f_3, \ldots, f_k]$ и монотонна. Новая программа действует аналогично прежней с той разницей, что на каждом шаге появляется степень числа P — лишь для того, чтобы быть немедленно стёртой до перехода к следующему шагу. □

Будем называть Фрактран-$1\frac{1}{2}$-программу

$$f_1^*, f_2^*, \ldots, f_j^*[f_1, f_2, \ldots, f_k]$$

монотонной, если

$$f_1^* < f_2^* < \ldots < f_j^*, \quad f_1 < f_2 < \ldots < f_k.$$

Тогда наша универсальная программа симулирует монотонные Фрактран-$1\frac{1}{2}$-программы. Она кодирует такую программу тремя числами M^*, M и d, которые определяются следующим образом.

Пусть d — какой-либо общий знаменатель всех упомянутых дробей, и пусть данная Фрактран-$1\frac{1}{2}$-программа имеет вид

$$\frac{m_1^*}{d} \quad \frac{m_2^*}{d} \quad \ldots \quad \frac{m_j^*}{d} \quad \left[\frac{m_1}{d} \quad \frac{m_2}{d} \quad \ldots \quad \frac{m_k}{d}\right].$$

Добавим теперь дополнительные числа m_{j+1}^* и m_{k+1}, кратные d и удовлетворяющие условиям

$$m_1^* < m_2^* < \ldots < m_j^* < m_{j+1}^*, \quad m_1 < m_2 < \ldots < m_k < m_{k+1}$$

и

$$\left[\frac{1}{2}M^*\right] \leqslant M,$$

где

$$M^* = 2^{m_1^*} + 2^{m_2^*} + \ldots + 2^{m_{j+1}^*}, \quad M = 2^{m_1} + 2^{m_2} + \ldots + 2^{m_{k+1}}.$$

Универсальная программа POLYGAME, стартующая с

$$2^N 3^M 5^{M^*} 17^{d-1} 23,$$

симулирует данную Фрактран-$1\frac{1}{2}$-программу, стартующую с N. Эта универсальная Фрактран-1-программа получена из Фрактран-программы с рис. 4. Соответственно, рассмотрим действие последней, когда она стартует из узла 23 при $r_2 = N$, $r_3 = M$, $r_5 = M^*$, $r_{17} = d - 1$.

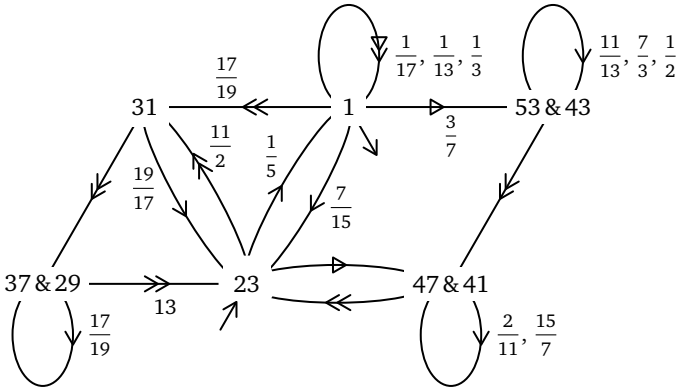


Рис. 4. Блок-схема для POLYGAME

Программа работает примерно следующим образом. Когда найдено новое N, программа вычисляет его последовательные кратные $N, 2N, 3N, \ldots, mN$ и при этом многократно уменьшает M вдвое, получая $[M/2], [M/4], \ldots, [M/2^m]$. Если $[M/2^m]$ нечётно, так что m равно одному из m_i, то программа проверяет, кратно ли Nm числу d. Если да, то программа обновляет M и берёт новое $N = mN/d$, *кроме случая* $m = m_{k+1}$ (т. е. $[M/2^m] = 1$), когда программа останавливается в узле 1, причём регистр 2 содержит N, а остальные регистры пусты. При первом проходе программа использует M^* вместо M.

Регистры 13, 17, 19 функционируют как счётчики, причём их значения загружаются в таком виде, что мы сразу можем видеть, кратны ли они d. Если $r_{13} = q$, $r_{19} = r$, $r_{17} = d-1-r$, $0 \leqslant r < d$, то *счёт* равен $qd + r$. Если программа приходит в узел 31 («входит в счётчик») при этих значениях и затем приходит в узел 23 («покидает счётчик»), то мы получаем

$$r_{13} = q, \quad r_{19} = r+1, \quad r_{17} = d-1-(r+1) \quad \text{при } r < d-1,$$
$$r_{13} = q+1, \quad r_{19} = 0, \quad r_{17} = d-1 \quad \text{при } r = d-1.$$

Иначе говоря, счёт увеличится на 1.

Таким образом, если программа стартует с 23 при $r_5 = r_{11} = 0$ и $r_2 = N$, то она увеличивает счёт на N при передаче N из регистра 2 в регистр 11, а затем идёт в узел 47 (где её первое действие — вернуть N из регистра 11 в регистр 2).

После этих замечаний читателю не составит трудности проверить переходы между конкретными конфигурациями, показанными в таблице 2.

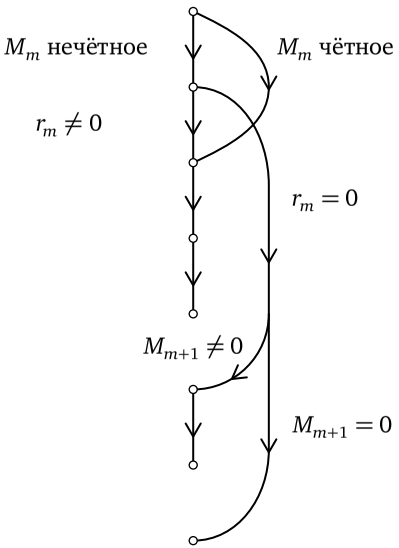
Пусть для некоторых положительных чисел d, N, M, M_0, удовлетворяющих неравенству $\left[\frac{1}{2}M_0\right] \leqslant M$, и переменных значениях m определены числа M_m, q_m, r_m по формулам

$$M_m = \left[\frac{M_0}{2^m}\right], \quad mN = q_m d + r_m \ (0 \leqslant r_m < d).$$

Тогда таблица 2 показывает, что, кроме случая нечётного M_m при $r_m = 0$, конкретная конфигурация из первой строки приводит к аналогичной конфигурации (из пятой строки), где m возросло на 1. В указанном особом случае, если $M_{m+1} \neq 0$, мы получаем другую подобную конфигурацию (в седьмой строке), но с нулевым значением m (и счёта), новым начальным значением $M_0 = M$ для M_m и новым значением mN/d для N. Если же M_{m+1} равно 0, мы попадаем на последнюю строку таблицы и *останавливаемся* в узле 1, причём в регистре 2

Таблица 2. Действие программы POLYGAME

Узел	Содержимое регистров: 2	3	5	7	11	13	17	19	Действие
23	N	M	M_m	0	0	q_m	$d-1-r_m$	r_m	
1	N	$M-M_{m+1}$	0	M_{m+1}	0	q_m	$d-1-r_m$	r_m	
23	N	$M-M_{m+1}$	0	M_{m+1}	0	q_m	$d-1-r_m$	r_m	
47 & 41	0	$M-M_{m+1}$	0	M_{m+1}	0	q_{m+1}	$d-1-r_{m+1}$	r_{m+1}	
23	N	M	M_{m+1}	0	0	q_{m+1}	$d-1-r_{m+1}$	r_{m+1}	
47 & 41	0	0	0	M	$\frac{mN}{d}$	0	$d-1$	0	
23	$\frac{mN}{d}$	M	M	0	0	0	$d-1$	0	
1	N	0	0	0	0	0	0	0	



$$mN = q_m \cdot d + r_m \ (0 \leqslant r_m < d), \quad M_m = [M_0/2^m]$$

стоит N, а остальные регистры пусты. В случае нечётного M_m и $r_m = 0$ мы говорим о *перезагрузке*.

Пусть теперь программа стартует с конкретной конфигурации в верхней строке таблицы, причём $m = 0$, а начальное значение M_0 величины M_m равно числу $2^{m_0} + 2^{m_1} + \ldots + 2^{m_{k+1}}$, где $m_0 < m_1 < \ldots < m_{k+1}$, причём m_{k+1} кратно d. Тогда до следующей перезагрузки выполнены эквивалентности

$$\begin{aligned} M_m \text{ нечётно} &\iff m \text{ — одно из } m_i, \\ r_m = 0 &\iff \frac{mN}{d} \text{ целое}, \\ M_{m+1} = 0 &\iff m = m_k. \end{aligned}$$

Значит, следующая перезагрузка будет при первом m_i, для которого $m_i N/d$ целое, и произойдёт либо *замена* N на $m_i N/d$, а также m на 0 и M_m на M (если $i < k$), либо *остановка* в узле 1, причём тогда в регистре 2 стоит N, а остальные регистры пусты ($i = k$).

На этом необходимые проверки заканчиваются. Вначале мы полагаем $m = 0$ и $M_0 = M^*$, но все последующие перезагрузки дают $M_0 = M$, в соответствии с правилами для Фрактран-$1\frac{1}{2}$-программ.

Фрактран-1-программа — это Фрактран-$1\frac{1}{2}$-программа с $M = M^*$. В этом случае можно использовать другое списочное число, а именно $7^M 17^{d-1} 41$.

§ 11. Применения, модификации, благодарности

Проблема Коллатца для функции

$$g(N) = \begin{cases} \frac{1}{2}N & (N \text{ чётно}), \\ 3N + 1 & (N \text{ нечётно}) \end{cases}$$

состоит в следующем: верно ли, что для любого натурального числа N существует такое k, что $g^k(N) = 1$? Обзор по этой проблеме см. в работе [3].

Аналогичные вопросы можно поставить для более общих *функций Коллатца* $g(N) = a_N N + b_N$, где a_N и b_N — рациональные числа, зависящие лишь от значения N по модулю фиксированного числа D. В работе [1] мы доказали, что не существует алгоритма для решения произвольных проблем Коллатца. Действительно, для любой вычислимой функции $f(n)$ существует Фрактран-1-программа $[f_1, f_2, \ldots, f_k]$ со следующим свойством: если стартовать в 2^n, то первая следующая

степень двойки равна $2^{f(n)}$. Другими словами, можно определить f формулой $2^{f(n)} = g^k(2^n)$, где k — наименьшее натуральное число, для которого $g^k(2^n)$ — степень двойки, и функция $g(N)$ вышеуказанного вида равна $f_i N$ для наименьшего i, при котором это число целое. Этот результат выражает в явном виде *теорему Клини о нормальной форме*.

Заметим, что $g(N)/N$ — периодическая функция с рациональными значениями, так что $g(N)$ — функция Коллатца, для которой всегда $b_N = 0$. Таким образом, даже для функций Коллатца этого специального типа невозможна разрешающая процедура. Применяя это рассуждение к универсальной игре с дробями, можно получить *конкретную* проблему типа Коллатца, не имеющую разрешающей процедуры.

(Отметим, что, разумеется, проблемы Коллатца с произвольным b_N труднее, а не легче решать. Например, можно сформулировать проблему, которая симулирует программу из 10 сегментов, причём в каждом сегменте используются лишь числа с заданной цифрой на последнем месте, а управление передаётся между сегментами лишь в определённые ключевые — и рекурсивно непредсказуемые — моменты.)

Джон Рикард сказал мне, что он нашёл универсальную программу с семью дробями, имеющую вид $2^{2^n} \cdot c \to 2^{2^{f(n)}}$, и программу с девятью дробями, имеющую вид $2^n \cdot c \to 2^{f(n)}$. Однако похоже, что его дроби намного сложнее того, что можно будет когда-нибудь явно выписать. В § 10 я применил одну из идей Рикарда[2)]. Майк Гай оказал ценную помощь в вычислении списочных чисел из § 2. Разумеется, ответственность за любые ошибки в этих числах полностью остаётся на нём.

Список литературы

[1] *Conway J. H.* Unpredictable iterations // Proc. Number Theory Conference. Boulder, Colorado, 1972. P. 49–52.

[2] *Conway J. H.* FRACTRAN — A simple universal programming language for arithmetic // Open Problems Commun. Comput. 1986. P. 4–26. (См. с. 91–110 наст. издания.)

[3] *Lagarias J. C.* The $3x + 1$ problem and its generalizations // Amer. Math. Monthly. 1985. V. 92, № 1. P. 3–25.

[4] *Mahler K.* On the approximation of π // Indagationes Math. 1953. V. 15. P. 30–42.

[2)] Недавно три школьника из Германии Леннарт Граббель, Юрий Каганский и Пауль Зиверт создали универсальную Фрактран-программу («Polygame») всего лишь из 13 дробей; она имеет вид $29 \cdot 2^{c \cdot 2^n} \to 2^{2^{f(n)}} \cdot r(n)$, где $r(n)$ — нечётный коэффициент, несущественный для результата (пока не опубликовано).