В. Журавлев, П. Самовол, Быстрее быстрого, или Можно ли обогнать бинарный алгоритм, *Квант*, 2013, номер 2, 7–15

Быстрее быстрого, или Можно ли обогнать бинарный алгоритм

В.ЖУРАВЛЕВ, П.САМОВОЛ

Чтобы преодолеть путь, нужно сначала преодолеть половину пути, а чтобы преодолеть половину пути, нужно сначала преодолеть половину половины, и так до бесконечности.

Зенон Элейский. Апория «Дихотомия»

Введение

Современный мир немыслим без калькуляторов и компьютеров. Эти средства вычислительной техники могут быстро складывать, умножать, возводить в степень и делать еще массу разнообразнейших операций. Однако быстрое возведение в степень сводится к быстрому умножению, а быстрое умножение – к быстрому сложению. Некоторые алгоритмы, которые позволяют ускорять вычисления, были известны еще древним.

В нашей статье, на примере нескольких задач, мы хотим сравнить различные алгоритмы и посмотреть, какой из них и в каких случаях работает быстрее. Некоторые из этих задач с простыми формулировками в общем случае оказались достаточно крепкими орешками. Мы будем формулировать задачи так, как они нам встречались для частных случаев, с соответствующей ссылкой на источник, и если в источнике не было обобщения, то отдельным пунктом мы будем формулировать соответствующее обобщение. Иногда задачи будут очень похожи и связаны между собой, однако все же у них будут определенные отличия. Для тех, кто не хочет останавливаться на достигнутом, мы предложим несколько новых задач для исследования.

Задачи

Задача 1

а) ([1]) Дан ящик сахарного песка, чашечные весы и гирька в 1 г. Как возможно быстрее отвесить покупателю 1 кг сахара? (Укажите схему уравновешиваний.)

б) Тем, кто предпочитает не метрическую систему мер и весов, мы предлагаем решить эту задачу для гирьки в 1 унцию и 100 фунтов сахара (1 фунт = 12 унций).

в) Как решить общую задачу по взвешиванию n г сахара с помощью чашечных весов и гирьки в 1 г?

Задача 2 (М1086, М.В.Сапир)

С числом разрешено производить две операции: «увеличить вдвое» и «увеличить на 1». За какое наименьшее число операций можно из числа 0 получить: а) 100; б) n, если сумма цифр двоичной записи числа n равна s?

Задача 3 [1] (М240, Э.Г.Белага)

По заданному x значение x^8 можно найти за три арифметических действия: $x^2 = x \cdot x$, $x^4 = x^2 \cdot x^2$, $x^8 = x^4 \cdot x^4$, а x^{15} – за пять действий: первые три – те же самые, затем $x^8 \cdot x^8 = x^{16}$ и $x^{16} : x = x^{15}$. Докажите, что:

а) x^{1000} можно найти за 12 действий (умножений и делений);

б) для любого натурального n значение x^n можно найти не более чем за $\frac{3}{2}\log_2 n + 1$ действий.

Следующая задача была опубликована в 1894 году во французском математическом журнале и до сих пор актуальна.

Задача 4 ([2], Х.Деллак)

Какое наименьшее число операций умножения необходимо для возведения числа x в степень n?

Задача 5

а) (фольклор) Есть два стеклянных шарика и 100-этажный дом. Вы бросаете шарик с разных этажей этого дома, чтобы выяснить, начиная с какого этажа шарик будет разбиваться от падения (например, падая с пятого, уже разбивается, а с четвертого – еще нет). Вопрос: какое минимальное количество бросков понадобится для того, чтобы точно узнать, с какого именно этажа шарики начинают разбиваться? Тот же вопрос в случае n-этажного дома.

б) Предположим, что шариков неограниченное количество. Какое минимальное число испытаний (бросков с этажей дома) наверняка хватит, чтобы определить крепость шарика (т.е. этаж, начиная с которого шарики начинают разбиваться). Тот же вопрос в случае n-этажного дома.

Задача 6

а) (Третья Соросовская олимпиада школьников, задача 10-II-6, В.Протасов)

Имеется 76 карточек, на которых написаны различные числа. Эти карточки разложены на столе по кругу числами вниз. Надо найти какие-нибудь три идущие подряд карточки такие, что число, написанное на средней из этих трех карточек, больше, чем на каждой из двух соседних. Перевернуть можно последовательно не более 10 карточек. Как надо действовать, чтобы наверняка найти три карточки, для которых выполняется указанное условие?

[1] В задачах 3 и 4 предполагается, что $x \neq 0$ и $x \neq 1$.

б) ([3], В.Протасов) По кругу растут 199 деревьев, все разного возраста. Можно ли выяснить возраст 12 деревьев так, чтобы наверняка найти дерево, старшее обоих своих соседей (слева и справа)?

в) По кругу растут n деревьев разного возраста, $n \geq 3$. У какого минимального числа деревьев необходимо выяснить возраст, чтобы наверняка найти дерево, старшее обоих своих соседей (слева и справа)?

Обсуждение задач

В каждой из этих задач каждому натуральному числу n мы можем сопоставить соответствующее минимальное число операций (действий, взвешиваний), тем самым получив для каждой задачи свою числовую последовательность. Фактически мы получаем некие алгоритмы для каждой задачи, при этом n-й член последовательности в точности равен числу шагов соответствующего алгоритма. Так, в условии задачи 3 приведены алгоритмы получения x^8 и x^{15}, соответственно, за 3 и за 5 шагов, а поскольку эти алгоритмы являются минимальными (проверьте), то восьмой член этой последовательности равен 3, а 15-й член последовательности равен 5.

Введем обозначения для последовательностей, которые у нас появились:

$w(n)$ – минимальное число взвешиваний, необходимое для получения n г сахара на чашечных весах с гирькой в 1 г (задача 1,в);

$t(n)$ – минимальное число операций, необходимое для получения числа n (задача 2,б);

$m(n)$ – минимальное число действий (умножений и делений), необходимое для возведения числа x в степень n (задача 3);

$l(n)$ – минимальное число умножений, необходимое для возведения числа x в степень n (задача 4);

$b(n)$ – минимальное число бросков в задаче с шариками (задача 5,б).

Выпишем в таблицу несколько начальных значений этих последовательностей:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$w(n)$	1	2	2	3	3	3	3	4	4	4	4	4	4	4	4	5
$t(n)$	1	2	3	3	4	4	5	4	5	5	6	5	6	6	7	5
$m(n)$	0	1	2	2	3	3	4	3	4	4	5	4	5	5	5	4
$l(n)$	0	1	2	2	3	3	4	3	4	4	5	4	5	5	5	4
$b(n)$	1	2	2	3	3	3	3	4	4	4	4	4	4	4	4	5

Как видно из таблицы, у первой последовательности есть явная закономерность: присмотритесь к членам последовательности, стоящим на 2-м, 4-м, 8-м, 16-м местах. Этих данных вполне достаточно, чтобы сформулировать предположение индукции.

Закономерность существует и для второй последовательности, хотя она и не так заметна. Посмотрите, например, на значения членов последовательности, стоящих на нечетных местах, и на значения членов последовательности, стоящих перед ними. Хотя, конечно, для предположения индукции необходимы дополнительные рассуждения.

Сравнив задачи 3 и 4, логично заключить, что операциями умножения и деления мы достигнем результата не медленнее, чем только операциями умножения, т.е. $m(k) \leq l(k)$. Как ни странно, но мы видим, что начальные члены последовательностей $m(n)$ и $l(n)$ совпадают. Тем не менее, существует бесконечно много чисел k, для которых $m(k) < l(k)$. Точные формулы для членов этих последовательностей не найдены, но все-таки можно оценить значения членов последовательности или, как мы говорили, число шагов соответствующего алгоритма.

Что же объединяет все эти задачи? По нашему мнению, ко всем этим задачам применим бинарный метод, или дихотомия[2]. Вероятно, правильно говорить о бинарном методе, когда мы что-то удваиваем, и о дихотомии, когда мы что-то делим пополам.

Рассмотрим пункт б) первой задачи. Очевидно, что мы можем взвесить еще столько же сахара, сколько лежит на одной из чашек весов. Таким образом, взвешиваем с помощью гирьки 1 унцию сахара, затем, убрав гирьку, взвешиваем на весах еще одну унцию сахара и ссыпаем сахар на одну чашу весов. Получаем на одной чаше весов 2 унции сахара, взвешиваем еще две унции сахара, ссыпаем на одну чашу, получаем уже четыре унции сахара. Повторяем эту процедуру, получая последовательно 8, 16, 32, 64, 128, 256, 512, 1024 унций сахара. Поскольку 100 фунтов = 1200 унций, а $1200 = 1024 + 128 + 32 + 16$, то мы можем управиться не более чем за $30 = 11 + 8 + 6 + 5$ взвешиваний. А если бы мы, имея 16 унций сахара, взвесили один раз 16 унций сахара и отсыпали бы их в другую емкость, а затем, взвесив во второй раз 16 унций, пересыпали бы их на другую чашу весов, то получили бы опять 32 унции на чаше весов плюс уже отмеренные 16 унций. Поступая точно так же с 32 унциями, а затем – со 128 унциями, мы управились бы за $14 = 11 + 1 + 1 + 1$ взвешиваний. Понятно, что это достаточно быстрый, хотя и не минимальный алгоритм. Все дело в том, что после первого взвешивания мы убрали гирьку и больше ее не использовали.

Использование гирьки в дальнейших взвешиваниях позволяет нам не только обойтись без дополнительной емкости, но и получить минимальный алгоритм взвешивания. Хитрость заключается еще и в том, что гирьку мы можем положить как на одну, так и на

[2] Дихотомия – от греч. διχοτομια. Слово составлено из διχη – надвое и τομη – деление.

другую чашу весов. Таким образом, если мы взвесили k унций сахара, то за следующее взвешивание мы можем получить на другой чаше весов $k-1$, k или $k+1$ унций сахара в зависимости от того, использовали ли мы гирьку, и если использовали, то на какую чашу весов ее положили. Ссыпав сахар вместе, мы получим на этом шаге $2k-1$, $2k$ или $2k+1$ унцию сахара соответственно. Именно из-за этой хитрости значения членов последовательности $w(n)$ от номера 2^q до номера $2^{q+1}-1$ все равны $q+1$. Другими словами,

$$w(n)=[\log_2 n]+1,$$

где $[x]$ – целая часть числа x. Мы оставляем читателю провести несложное доказательство этого факта при помощи метода математической индукции. Поскольку $2^{10}=1024<1200<2047=2^{11}-1$, то минимальное число взвешиваний для 100 фунтов сахара равно 11.

Посмотрим на вторую задачу через призму первой. Во-первых, нетрудно понять, что $t\left(2^q\right)=q+1$. Во-вторых, по условию из числа k на следующем шаге мы можем получить либо $k+1$, либо $2k$. На этом сходство с первой задачей вроде бы заканчивается.

Попробуем все-таки решить пример. Будем действовать с конца. Применим к полученному на каком-то шаге числу операции, обратные к имеющимся в условии, т.е. будем делить на 2 и вычитать 1. Мы легко построим цепочку:

$$100\xrightarrow{1/2}50\xrightarrow{1/2}25\xrightarrow{-1}24\xrightarrow{1/2}$$
$$12\xrightarrow{1/2}6\xrightarrow{1/2}3\xrightarrow{-1}2\xrightarrow{1/2}1\xrightarrow{-1}0.$$

Теперь остается вновь обратить операции, и мы получим искомую минимальную цепочку:

$$0\xrightarrow{+1}1\xrightarrow{\times2}2\xrightarrow{+1}3\xrightarrow{\times2}6\xrightarrow{\times2}12$$
$$12\xrightarrow{\times2}24\xrightarrow{+1}25\xrightarrow{\times2}50\xrightarrow{\times2}100. \quad (*)$$

Вот он, бинарный метод, в действии!

Понятно, что если мы действуем с конца и у нас получается нечетное число, то мы можем только вычесть 1. Если же мы получаем четное число, то, раз мы ищем минимальную цепочку, нам необходимо делить на 2, поскольку при вычитании единицы число шагов будет больше. Строгое доказательство этого факта также можно провести методом математической индукции.

Однако во второй задаче нас интересует подсказка, которая фигурирует в самом условии задачи, а именно фраза «сумма цифр двоичной записи числа n». Дело в том, что не только двоичная запись числа, но и параметр «сумма цифр двоичной записи» играют одну из ключевых ролей не только в этой, но и в следующих задачах. Но об этом чуть позже.

В первых двух задачах мы умножали на 2, добавляли или вычитали 1, а в третьей и четвертой задаче мы возводим числа в степень. Неужели существует связь между этими задачами? Да: дело в том, что основание степени мы нигде не используем, и все операции происходят с показателями.

Так, пример получения из x значения x^8 из задачи 3 аналогичен получению из 1 числа 8 при помощи удвоения. Рассмотрим построенную выше цепочку $(*)$, начиная с 1. Мы получим следующий быстрый алгоритм возведения в 100-ю степень:

$$x\cdot x=x^2,\ x^2\cdot x=x^3,\ x^3\cdot x^3=x^6,\ x^6\cdot x^6=x^{12},$$
$$x^{12}\cdot x^{12}=x^{24},\ x^{24}\cdot x=x^{25},\ x^{25}\cdot x^{25}=x^{50},$$
$$x^{50}\cdot x^{50}=x^{100}.$$

Казалось бы, за исключением «сдвига» (в задаче 2 операции начинаются с 0, а в задачах 3 и 4 операции начинаются с 1), мы имеем нужный нам алгоритм для решения задачи 4. Чтобы устранить «сдвиг», достаточно рассмотреть последовательность $t(n)-1$, соответствующий алгоритм которой ровно на один шаг короче, чем у последовательности $t(n)$. Для начальных значений нужно вычесть в соответствующей строке таблицы по 1 из каждой клетки этой строки. Чтобы увидеть разницу между задачами, посмотрим, как получить из x значение x^{15}.

Бинарный метод нам дает

$$15\xrightarrow{-1}14\xrightarrow{1/2}7\xrightarrow{-1}6\xrightarrow{1/2}3$$
$$3\xrightarrow{-1}2\xrightarrow{1/2}1 \text{ и обратно}$$
$$1\xrightarrow{\times2}2\xrightarrow{+1}3\xrightarrow{\times2}6\xrightarrow{+1}7\xrightarrow{\times2}14$$
$$14\xrightarrow{+1}15 \text{ – всего 6 шагов.}$$

В терминах возведения в степень имеем следующий быстрый алгоритм:

$$x\cdot x=x^2,\ x^2\cdot x=x^3,\ x^3\cdot x^3=x^6,\ x^6\cdot x=x^7,$$
$$x^7\cdot x^7=x^{14},\ x^{14}\cdot x=x^{15}.$$

Но мы можем пойти и по-другому, а именно так: $x\cdot x=x^2$, $x^2\cdot x=x^3$, $x^3\cdot x^3=x^6$, $x^6\cdot x^6=x^{12}$, $x^{12}\cdot x^3=x^{15}$;

или так: $x\cdot x=x^2$, $x^2\cdot x=x^3$, $x^3\cdot x^2=x^5$, $x^5\cdot x^5=x^{10}$, $x^{10}\cdot x^5=x^{15}$.

Эти алгоритмы состоят из 5 шагов! Отметим, что мы получили x^{15} за 5 шагов только одними умножениями без деления (сравните с примером задачи 3). Следовательно, бинарный метод является *быстрым*, но не всегда *минимальным* алгоритмом.

Искушенный читатель отметит, что, несмотря на сходство задачи 2 с задачами 3 и 4, между ними есть важное, хотя и не заметное на первый взгляд отличие. Это отличие легко понять любому школьнику, знакомому с программированием. Ведь в задаче 2 мы не запоминаем промежуточные результаты вычислений, а в задачах 3 и 4 промежуточные результаты вычислений используются нами неоднократно. Поэтому можно сказать, что уменьшение количества операций при использовании соответствующих алгоритмов в задачах 3 и 4 достигается за счет использования большей памяти.

В предыдущем примере для быстрого возведения в степень мы применили так называемый *метод множителей*. Обе короткие цепочки из примера построены исходя из того, что $3\cdot5=5\cdot3=15$. Если вспомнить

свойство степеней $x^{mk} = (x^m)^k = (x^k)^m$, то становится понятным второй алгоритм построения цепочек. Если мы имеем разложение на отличные от 1 множители $n = mk$, то сначала мы находим x^m, а затем возводим результат в степень k. Если же n – простое число, то мы таким способом находим x^{n-1} и на последнем шаге умножаем на x. В зависимости от n, лучшим оказывается то один, то другой метод. Оба эти алгоритма могут быть «модернизированы», и, конечно, их можно скомбинировать. Известен и еще один метод, но об этом чуть ниже.

В настоящее время в общем виде задача нахождения минимального алгоритма для быстрого возведения в степень не решена и вполне может рассматриваться как задача для серьезного исследования. В то же время, разными математиками получен ряд результатов, которые мы хотим сообщить и которые могут рассматриваться как самостоятельные задачи и упражнения, поскольку их решение вполне по силам школьникам и студентам.

Как мы уже говорили выше, основание степени нигде не используется, поэтому мы можем переформулировать задачи 3 и 4 в терминах сложения и вычитания (умножение на 2 – это прибавление к числу самого себя). Вот что получится.

1) Калькулятор может только складывать числа, имеющиеся в его памяти. Первоначально в памяти калькулятора забита 1 (единица), количество ячеек памяти не ограничено. Спрашивается, за какое минимальное число операций из 1 можно получить n?

2) Калькулятор может складывать или вычитать числа, имеющиеся в его памяти. Первоначально в памяти калькулятора забита 1 (единица), количество ячеек памяти не ограничено. Спрашивается, за какое минимальное число операций из 1 можно получить n?

Отметим, что если задачу 2 тоже переформулировать на языке калькуляторов, то у такого калькулятора будет всего две ячейки памяти, в одну из которых всегда забита 1.

Упражнения

1 ([4], Н. Агаханов). Число, написанное на доске, каждую минуту либо удваивается, либо из него вычитается единица. После нескольких таких операций из числа 1 было получено число 2002. Докажите, что в некоторый момент на доске было записано число, содержащее в своей записи цифру 3.

2 [5]. Компьютерная программа преобразует набор из натуральных чисел по следующему правилу: каждое четное число делится на два, а из каждого нечетного числа вычитается единица. Докажите, что если начальный набор состоял из пяти последовательных натуральных чисел, то после двух преобразований в нем появятся хотя бы два равных числа.

Однако вернемся к двоичной записи числа n. Пусть

$$n = b_q \cdot 2^q + b_{q-1} \cdot 2^{q-1} + \ldots + b_1 \cdot 2 + b_0,$$

где $b_q = 1$, $b_i \in \{0,1\}$.

Обозначим через $\lambda(n) = q$ уменьшенную на 1 длину двоичной записи числа n, а через $\nu(n)$ – сумму цифр, т.е. количество тех чисел b_i, которые равны 1. Понятно, что $\lambda(n) = q = [\log_2 n]$. Теперь мы можем легко получить ответ к задаче 2 для любого числа n, а именно с учетом «сдвига»:

$$t(n) = \lambda(n) + \nu(n) = [\log_2 n] + \nu(n).$$

Как мы отмечали, задача 4 оказалась в общем случае «крепким орешком», и вокруг нее образовалось серьезное поле для исследований. Мы приведем без доказательства несколько теорем, которые показывают, как развивались и усовершенствовались быстрые алгоритмы. Теоремы и их доказательства, а также некоторые упражнения можно найти в [7]. Для начала все же посмотрим на несколько частных примеров.

Если $\nu(n) = 1$, то n является степенью двойки. Поскольку $l(n)$ – это число шагов в минимальном алгоритме, то $l(2^q) = q$. Это равенство доказывается по индукции и с учетом сдвига согласуется с приведенным ранее равенством $t(2^q) = q + 1$. Если $\nu(n) = 2$, т.е. n является суммой двух различных степеней двойки, то $l(n) = l(2^q + 2^r) = q + 1$ для $q > r$.

Если $\nu(n) = 3$, т.е. n является суммой трех различных степеней двойки, то выполняется следующая

Теорема 1. $l(2^q + 2^r + 2^p) = q + 2$ *для* $q > r > p$.

Доказательство этой теоремы не лежит на поверхности и требует детального рассмотрения возможных вариантов.

Из этих примеров наглядно видно, что бинарный метод является оптимальным, когда $\nu(n) \le 3$. Более того, непосредственное рассмотрение свойств бинарного метода позволяет нам получить оценки на $l(n)$ для общего случая. Эти оценки сформулированы в следующей теореме.

Теорема 2. *Выполняются неравенства* $\lambda(n) \le l(n) \le \lambda(n) + \nu(n) - 1$.

Доказательство теоремы 2 вполне элементарно, и мы оставляем его читателям в качестве упражнения.

Упражнение 3. Докажите неравенство $l(n) \ge \log_2 n$ и приведите примеры, когда оно обращается в равенство.

Мы уже видели, что при $m = 15$ бинарный метод не оптимален. Нетрудно проверить, что он не оптимален для $m = 23$ или $m = 39$ (в этих частных случаях $\nu(n) = 4$). Исследования показывают, что при возрастании числа $\nu(n)$ количество возможных случаев существенно увеличивается. Это иллюстрирует следующая теорема, в которой рассмотрен случай $\nu(n) = 4$, еще поддающийся разумному описанию.

Теорема 3 (Д.Кнут). *Если* $\nu(n) \ge 4$, *то* $\lambda(n) + 3 \le l(n)$ *за исключением следующих четырех случаев а)–г), для которых* $l(2^q + 2^r + 2^p + 2^t) = q + 2$, *где* $q > r > p > t$:

а) $q - r = p - t$ *(пример:* $n = 15$*);*
б) $q - r = p - t + 1$ *(пример:* $n = 23$*);*
в) $q - r = 3$, $p - t = 1$ *(пример:* $n = 39$*);*
г) $q - r = 5$, $r - p = p - t = 1$ *(пример:* $n = 135$*).*

Улучшение верхней оценки, полученной в теореме 2, достигается за счет рассмотрения m-арного метода. Мы

вскользь упоминали, что такой алгоритм является «модернизацией» бинарного метода.

Теорема 4 (А.Брауэр). *Для всех* $k \ge 1$ *выполняется неравенство* $l(n) \le (1+1/k)(\lambda(n)+1)+2^k$.

Упражнение 4. Используя оценки, полученные в теоремах 2 и 4, докажите, что

$$\lim_{n\to\infty} \frac{l(n)}{\log_2 n} = 1 .$$

Если рассмотреть метод множителей, то мы получим следующую теорему.

Теорема 5 (Де Жонкиэр). *Выполняется неравенство* $l(nm) \le l(n)+l(m)$.

Надеемся, что доказательство этой теоремы не составит труда для читателей. Оно следует из построения разобранного выше примера.

Если взять частный случай предыдущей теоремы для $m = 2$, то нетрудно видеть, что $l(2n) \le l(n)+1$. А если немного покопаться с числами, то кажется, что здесь на самом деле выполнено равенство, ведь интуитивно ясно, что быстрее удвоения ничего быть не может. Тем не менее, это не так! Существуют такие числа, что $l(2n) = l(n)$. Более того, как ни парадоксально, но совсем недавно (см. [8]) с помощью компьютеров было вычислено, что для $n = 375494703$ выполнено $34 = l(2n) < l(n) = 35$. Существование таких чисел n и m, что $l(nm) < l(n)$, вы можете попытаться проверить сами, решив упражнение 5.

Упражнения

5. Проверьте, что для чисел $n = \dfrac{2^{13}+1}{3} = 2731$ и $m = 3$ выполнено неравенство $l(nm) < l(n)$.

6. Приведите примеры, подтверждающие, что иногда метод множителей лучше, чем бинарный метод, а иногда наоборот. Покажите, что таких примеров бесконечно много.

Возможность скомбинировать бинарный метод и метод множителей применяется в доказательстве следующей теоремы.

Теорема 6 (В.Хансен). $l(2^q+km) \le q+\nu(k)+$ $+\nu(m)-1$, *если* $\lambda(k)+\lambda(m) \le q$.

Мы упоминали еще об одном методе. Визуально этот метод отличается от бинарного метода и метода множителей. Связано это с построением графов, являющихся деревьями. Тем не менее, такие построения нетрудно программируются. Можно сказать, что эти деревья приносят неплохие плоды.

Построим дерево, основанное на бинарном алгоритме. Принцип построения этого дерева объясним, как и ранее, «с конца». Отметим числами от 1 до n различные точки на плоскости; они будут вершинами графа. Теперь возьмем любую точку k, $k > 1$, и соединим ее ребром с точкой $k-1$, если k нечетное, и с точкой $k/2$, если k четное. В результате таких построений мы получаем граф, являющийся деревом. На рисунке 1 показано дерево для $n = 32$. Красным цветом мы показали путь на дереве, соответствующий одной из рассмотренных выше цепочек:

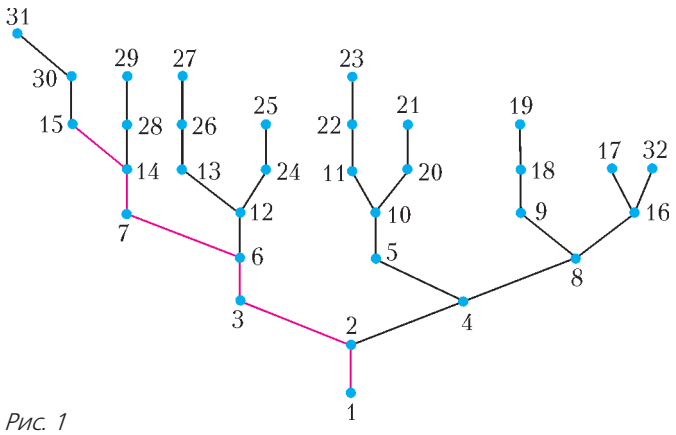

Рис. 1

$$15 \xrightarrow{-1} 14 \xrightarrow{1/2} 7 \xrightarrow{-1} 6 \xrightarrow{1/2} 3 \xrightarrow{-1} 2 \xrightarrow{1/2} 1 .$$

Построим также другое дерево немного иной формы (рис.2). Поуровневый алгоритм построения этого дерева будет сложнее. Первый уровень начинается с 1. Предположим, что мы построили k-й уровень дерева, построим теперь $(k+1)$-й уровень. Будем брать вершины по очереди, начиная с вершины с самым *большим* номером и далее по убыванию. Пусть мы дошли по очереди до вершины с номером m на k-м уровне, и при этом последовательность вершин $m = a_{k-1},\ldots,a_2,a_1,1$ будет обозначать путь от нашей вершины до корня дерева (вершины с номером 1). Теперь с $(k+1)$-го уровня присоединим к вершине с номером m только те вершины с номерами $2m = m+a_{k-1},\ldots,m+a_2,m+a_1,m+1$, которых еще нет в дереве.

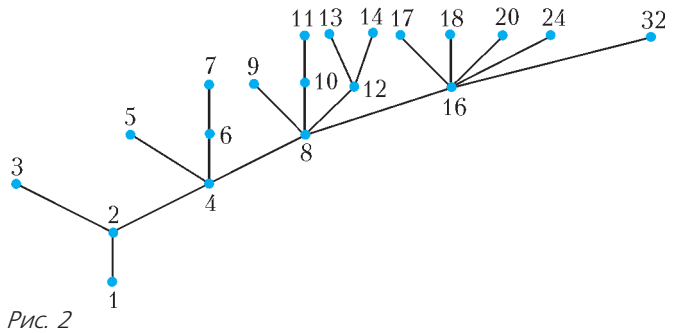

Рис. 2

Нетрудно видеть, что это дерево дает еще один метод возведения в степень. Заметим, что длина цепочки по этому алгоритму совпадает с длиной цепочки, вычисленной бинарным методом. Попробуйте доказать это.

Оказывается, что небольшие изменения в построении дерева приводят к методу, который работает быстрее бинарного метода и метода множителей на большом массиве чисел. Так, например, первое значение, на котором метод множителей быстрее метода дерева степеней, это $n = 19879$. Понятно, что такие вычисления могут быть проделаны только благодаря компьютерам.

Сформулируем алгоритм построения дерева для *метода дерева степеней.* Будем рассматривать это дерево как настоящие математики, т.е. перевернем его корнем вверх. Итак, как и прежде, первый уровень начинается с 1. Если k-й уровень дерева построен, то будем строить $(k+1)$-й уровень дерева. Вершины с k-го уровня берем по очереди, начиная с вершины с *наименьшим* номером и далее по возрастанию номеров. Пусть мы дошли по очереди до вершины с номером m на k-м уровне, и при этом последовательность вершин $m = a_{k-1},\ldots,a_2,a_1,1$ будет обозначать путь от корня дерева (вершины с номером 1) до нашей вершины. Теперь с $(k+1)$-го уровня присоединим к вершине с номером m только те вершины с номерами $2m = m + a_{k-1},\ldots,m + a_2, m + a_1, m + 1$, которых еще нет в дереве. На рисунке 3 построено дерево степеней до 7-го уровня.

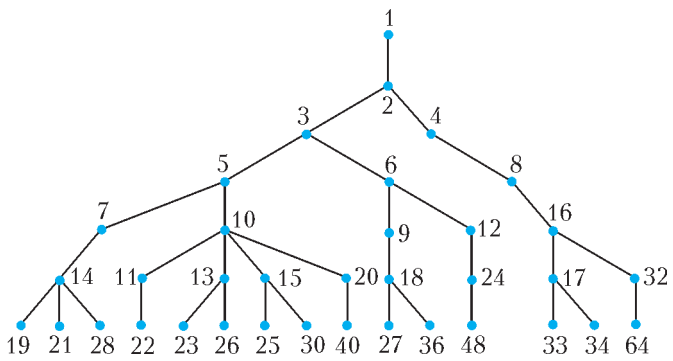

Рис. 3

Наличие таких разнообразных алгоритмов и применение комбинированных методов пока не дало окончательного решения этой, казалось бы простой, задачи. И даже в частном случае существует нерешенная *гипотеза Шольца–Брауэра*, утверждающая, что

$$l\left(2^n - 1\right) \le n - 1 + l(n).$$

Поскольку число $2^n - 1$ имеет максимальное количество единиц в двоичном представлении, то этот частный случай интересен тем, что является наихудшим случаем для бинарного метода. Вычисления на компьютерах показали, что для $1 \le n \le 64$ имеет место равенство (см. [7]).

Рассмотренные выше алгоритмы можно представить в наглядных терминах аддитивных цепочек. Однако мы не будем делать этого в рамках этой статьи. Пытливый читатель найдет информацию, относящуюся к аддитивным цепочкам, в литературе, приведенной ниже (см. [7–10]).

В нашем обсуждении мы практически не коснулись задачи 3. Дело в том, что в большинстве случаев при построении оптимального алгоритма нахождения степени при помощи умножений и делений можно построить алгоритм, который справляется только умножениями за то же число шагов. Выше мы построили пример без делений для получения x^{15} за 5 шагов. Предлагаем читателю построить 12-шаговый алгоритм получения x^{1000} с помощью умножения и деления и 12-шаговый алгоритм получения x^{1000} только с помощью умножения. Оценка, приведенная в пункте б) этой задачи, гораздо более слабая, чем оценка в теореме 4. Справедливости ради, стоит отметить, что существуют примеры, когда при помощи умножений и делений возведение в степень получается за меньшее число шагов, чем когда используются только умножения. Это, например, уже упоминавшееся в гипотезе Шольца–Брауэра число $2^n - 1$. С помощью умножений и делений получить число можно не более чем за $n + 1$ шаг.

Упражнения

7. Приведите еще примеры, когда при помощи умножений и делений возведение в степень получается за меньшее число шагов, чем когда используются только умножения.

8 ([5], Н. Агаханов). На доске написаны многочлены $x + 1$ и $x^2 + 1$. Разрешается дописывать на доску многочлен f, равный сумме, разности или произведению любых двух различных из написанных многочленов, если многочлен f не был написан на доске ранее. Можно ли написать на доске многочлен $x^{2006} + 1$?

Отвлечемся от темы степеней и посмотрим на задачу 5. Начнем с конца, т.е. с пункта б). В этом месте читатель должен был бы воскликнуть: «Бинарный метод!» Предлагаем читателю справиться с решением этого пункта, а точнее, найти полную аналогию с задачей 1 про гирьку и сахарный песок.

Пункт а) готовит нам сюрприз. Мы, конечно, можем применить бинарный метод и сбросить стеклянный шарик с 50-го этажа. И в случае если он разобьется, т.е. при неблагоприятном исходе, мы будем вынуждены провести еще 49 проверок. Мы не можем больше рисковать оставшимся шариком и применять бинарный метод дальше, поэтому придется бросать оставшийся шарик с 1-го этажа, со 2-го и так далее – возможно, до 49-го этажа. Выходит, при самом неблагоприятном исходе мы вынуждены затратить 50 попыток. Естественно, возникают подозрения в оптимальности такого алгоритма.

Любитель десятичной системы счисления предложит сбрасывать первый шарик с 10-го, 20-го, …, 90-го, 100-го этажей, а вторым шариком проверять оставшийся интервал в девять этажей. Такой алгоритм в самом неблагоприятном случае справится с поставленной задачей уже за 19 попыток. И хотя получился не оптимальный алгоритм, но стало понятно, что необходимо делать.

На самом деле, ответ для здания в 100 этажей – 14 попыток. Предоставим читателю в качестве упражнения найти соответствующую последовательность этажей. Остается сделать небольшую ремарку: ответ, который получился для здания в 100 этажей, подойдет и для зданий высотой в 101–105 этажей.

Варьируя число этажей, число шариков и даже число попыток, мы получим хорошую задачу для исследования, являющуюся обобщением задачи 5.

Упражнения

9. Есть B одинаковых стеклянных шариков и небоскреб. Вы можете бросать шарик с разных этажей небоскреба, чтобы выяснить, начиная с какого этажа шарик начинает разбиваться от падения (например, начиная с пятого уже

разбивается, а с четвертого – еще нет). В целях экономии средств вам заранее установлено постоянное число T таких тестов. Укажите оптимальный алгоритм проверки и найдите максимальный номер этажа небоскреба, который можно наверняка проверить.

10 ([6], С.Токарев). Мишень «бегущий кабан» находится в одном из n окошек, расположенных в ряд. Окошки закрыты занавесками так, что для стрелка мишень все время остается невидимой. Чтобы поразить мишень, достаточно выстрелить в окошко, в котором она в момент выстрела находится. Если мишень находится не в самом правом окошке, то сразу после выстрела она перемещается на одно окошко вправо; из самого правого окошка мишень никуда не перемещается. Какое наименьшее число выстрелов нужно сделать, чтобы наверняка поразить мишень?

Что делает задача 6 во всей этой истории, не совсем понятно. Тем не менее, формулировка вопроса «Как надо действовать, чтобы наверняка найти три карточки, для которых выполняется указанное условие?» подразумевает, что мы должны указать некоторый алгоритм действий, который приведет за некоторое количество шагов (не более 10 шагов в пункте а) и не более 12 шагов в пункте б) к требуемому результату.

Сразу отметим, что числа 76 и 199, определяющие количество карточек и деревьев, были подобраны не случайно. Автор задачи построил соответствующий алгоритм нахождения тройки предметов с требуемой упорядоченностью. Тем не менее, эти числа занижены. Мы утверждаем, что даже в случае, когда 89 карточек расположены по кругу, мы сможем найти не более чем за 10 шагов три идущие подряд карточки, удовлетворяющие условиям. Аналогично для случая, когда по кругу стоят 233 дерева: не более чем за 12 шагов нам удастся найти дерево, старшее своих соседей. Читатель, знакомый с последовательностью Фибоначчи, может определить, что 89 – это одиннадцатый член последовательности Фибоначчи, а 233 – это тринадцатый член этой последовательности. Пусть F_m обозначает m-й член последовательности Фибоначчи, где $F_{m+2} = F_{m+1} + F_m$, $F_1 = F_2 = 1$. Вот несколько начальных членов этой последовательности: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, … Теперь видно, что $76 = 21 + 55 = F_8 + F_{10} < F_{11} = 89$ и $199 = 55 + 144 = F_{10} + F_{12} < F_{13} = 233$.

Обозначим через $h(n)$ минимальное число шагов алгоритма, которое необходимо, чтобы наверняка (в самой неблагоприятной ситуации) найти три рядом расположенных предмета (карточек или деревьев), удовлетворяющих условиям задачи 6. Чтобы сразу доказать сформулированные в предыдущем абзаце утверждения, решим следующее упражнение.

Упражнение 11. По кругу растут $n = F_m$ деревьев разного возраста, $n \geq 3$. За один шаг разрешается узнать возраст одного дерева. Постройте алгоритм, позволяющий находить дерево, которое старше обоих своих соседей (слева и справа), не более чем за $m - 1$ шагов. Другими словами, докажите, что $h(F_m) \leq m - 1$.

Обсудим доказательство этого упражнения. Представим деревья точками, расположенными на окружности длины $n = F_m$ в вершинах некоторого правильного n-угольника. Нетрудно заметить, что длина дуги между соседними точками равна 1. Узнав возраст дерева, будем помечать соответствующую точку числом.

Теперь попытаемся избавиться от окружности. На первом шаге узнаем возраст любого дерева и отметим соответствующую точку окружности этим числом. Обозначим это число через a. На втором шаге от точки, помеченной числом a, пройдем по окружности по часовой стрелке[3] расстояние F_{m-1}. Мы попадем в точку, соответствующую некоторому дереву, и узнаем его возраст. Пометим эту точку числом b. Другими словами, длина дуги между точкой, помеченной числом a, и точкой, помеченной числом b, равна F_{m-1}. В точке с меньшим значением разрежем окружность и выпрямим ее в отрезок длины F_m. Окружность исчезла! И мы можем производить наши построения на отрезке целочисленной прямой.

Не теряя общности, можем считать, что $a < b$. Таким образом, мы проводим разрезание окружности в точке, помеченной числом a. Получаем отрезок с расположенными на нем точками (деревьями) через равные расстояния длины 1. Концы отрезка помечены одинаковыми числами, и одна из точек внутри отрезка помечена числом большим, чем числа на концах отрезка. Расстояние от помеченной точки до одного конца отрезка равно F_{m-1}, а расстояние до другого конца отрезка равно $F_{m-2} = F_m - F_{m-1}$.

Теперь нетрудно провести индукционные построения.

Предположение индукции.

Рассмотрим целочисленный отрезок длины F_k со следующими свойствами:

1) концы отрезка помечены числами x, y;

2) внутри отрезка есть точка, помеченная числом z, при этом $x < z$, $y < z$;

3) расстояние от точки, помеченной z, до одного конца отрезка равно F_{k-1}, а до другого конца отрезка равно $F_{k-2} = F_k - F_{k-1}$.

Тогда можно найти дерево, старшее обоих своих соседей, и соответственно помеченную точку, значение которой больше, чем значения соседних точек, не более чем за $k - 3$ шага. (С учетом двух первых шагов для разрезания, общее число шагов для окружности составит $k - 1 = k - 3 + 2$.)

Основание индукции для $n = 3 = F_4$ выполнено. Действительно, в этом случае остается сделать один шаг, узнав значение оставшейся на отрезке точки w (рис.4), при этом если $w < z$, то искомой будет точка z, если $w > z$, то искомой будет точка w.

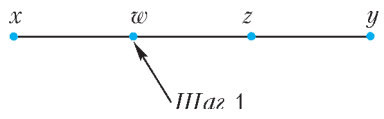

Рис. 4

Допустим теперь, что индукционное предположение выполнено для случая $k = m - 1$, т.е. чтобы отыскать требуемое дерево на отрезке с заданными свойствами длины F_{m-1}, нам нужно не более $m - 4$ шагов. Докажем, что индукционное предположение выполняется для $k = m$. Мы имеем ситуацию,

[3] Это направление выбрано для определенности.

которая изображена на рисунке 5. За один шаг мы проверяем возраст дерева и помечаем числом w точку, разбивающую отрезок AC длины F_{m-1} на части

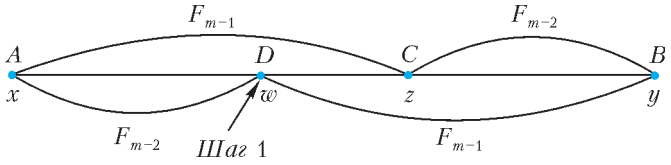

Рис. 5

длины F_{m-2} и $F_{m-3} = F_{m-1} - F_{m-2}$. Если $w < z$, то отрезок DB имеет длину F_{m-1} и для него выполнено индукционное предположение. Если $w > z$, то индукционное предположение выполнено для отрезка AC, также имеющего длину F_{m-1}. При этом мы затрачиваем не более $m - 3 = m - 4 + 1$ шагов.

Итак, методом математической индукции наше утверждение доказано.

Следовательно, с учетом двух первых шагов для разрезания окружности, общее число шагов для определения дерева, старшего своих соседей, составит не более $m - 1 = m - 3 + 2$ шагов.

Упражнение 12. Постройте алгоритм, который находит дерево, старшее своих соседей, для любого n, удовлетворяющего условиям $F_{m-1} < n \le F_m$, не более чем за $m - 1$ шаг. Другими словами, докажите, что $h(n) \le m - 1$.

Как видно из приведенных выше построений, нам удалось получить оценку сверху на минимальное число шагов алгоритма $h(n)$. По всей видимости, если $F_{m-1} < n \le F_m$, то имеет место равенство $h(n) = m - 1$. Тем не менее, в настоящий момент авторам статьи неизвестно доказательство этого факта.

Конечно, кроме алгоритма, основанного на числах Фибоначчи, мы можем предложить и другие алгоритмы для нахождения дерева, которое старше обоих своих соседей.

Еще раз посмотрим на общие принципы построения таких алгоритмов.

Как и ранее, на первых двух шагах мы узнаем возраст двух некоторых деревьев и помечаем соответствующие точки окружности числами. Затем проводим операцию разрезания окружности в точке с меньшим значением и выпрямляем окружность в отрезок.

Пусть на каком-то шаге у нас получился отрезок со следующими свойствами:

– концы отрезка помечены числами a, b;

– внутри отрезка есть точка, помеченная числом c, при этом $a < c$, $b < c$.

Если на следующем шаге нам удастся найти (построить) отрезок с аналогичными свойствами, но меньшей длины, то понятно, что через конечное число таких шагов этот процесс завершится построением отрезка длины 2 и нахождением трех точек (деревьев), удовлетворяющих условию задачи.

Остается выполнить этот шаг. Если длина отрезка $[a;b]$ больше 2, то возьмем один из отрезков $[a;c]$ или $[c;b]$, узнаем возраст дерева и пометим некоторую точку этим числом d. Не теряя общности, предположим, что эта точка лежит на отрезке $[a;c]$. Если $d < c$, то отрезок $[d;b]$ с помеченной точкой c имеет меньшую длину и обладает необходимыми свойствами. Если $d > c$, то необходимыми свойствами обладает отрезок $[a;c]$ с помеченной точкой d.

При таком построении мы можем рассмотреть уже упоминавшийся бинарный алгоритм. И хотя нам не всегда удастся выбирать очередную точку ровно посередине отрезка, в силу того что деревья расположены в целочисленных точках отрезка, мы можем выбирать эту точку как можно ближе к его середине. Мы можем получить и другие пропорциональные алгоритмы, если будем делить отрезок не пополам, а придерживаться какой-то пропорции. Так какой же из этих алгоритмов самый быстрый? Подробный ответ на этот вопрос нам неизвестен.

Наши дальнейшие рассуждения не будут строгим доказательством, а будут представлять некоторую приблизительную схему исследования этого вопроса. Попытаемся оценить скорость действия алгоритмов при самых неблагоприятных условиях. Откуда здесь возникли неблагоприятные условия? Дело в том, что при таких построениях, в зависимости от расположения точки d, длины отрезков $[a;c]$ и $[d;b]$ могут быть разными и поэтому скорость достижения нашей цели также может варьироваться. Понятно, что здесь имеется определенное сходство с задачей 5.

Посмотрим на несколько шагов действия бинарного алгоритма и алгоритма с коэффициентом пропорции k (рис.6). Пусть длина исходного отрезка равна l. Для построения пропорционального алгоритма возьмем

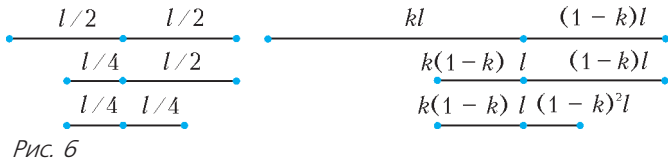

Рис. 6

$k > 1 - k$; при этом должно выполнятся неравенство $kl \le k(1-k)l + (1-k)l$. Такие построения возможны, если $\frac{1}{2} < k \le \frac{\sqrt{5}-1}{2}$.

Кажется, что после первого шага бинарный алгоритм действует быстрее, однако посмотрим, что происходит после двух шагов. После двух шагов бинарный алгоритм дает отрезок длины $l/2 = l/4 + l/4$. Пропорциональный алгоритм после двух шагов дает отрезок длины $(1-k)l = k(1-k)l + (1-k)^2 l$. Заметим, что $\left(1 - \frac{\sqrt{5}-1}{2}\right)l < \frac{l}{2}$. Это означает, что при значениях k, близких к $\frac{\sqrt{5}-1}{2}$, пропорциональный алгоритм будет действовать быстрее бинарного алгоритма. Причем самым быстрым был бы пропорциональный алгоритм золотого сечения. Известно, что наилучшее рациональное приближение к золотому сечению достигается при помощи дробей, числители и знаменатели которых являются последовательными числами Фибоначчи. Если в приведенных рассуждениях мы сделаем поправки на то, что имеем дело с целочисленными точками, то

рассмотренный выше алгоритм, основанный на числах Фибоначчи, выглядит вполне закономерно. Повторим, что приведенные выше построения с золотым сечением и числами Фибоначчи являются только некоторой схемой рассуждений и никак не претендуют на строгость.

Конечно, для небольших значений n можно написать соответствующую компьютерную программу, которая с помощью перебора вариантов подтвердит или опровергнет равенство $h(n) = m - 1$ при $F_{m-1} < n \le F_m$.

Упражнения

13. Проверьте, что для $n = 8$ бинарный алгоритм приводит к цели за 6 шагов (необходима проверка возраста 6 деревьев), в то время как алгоритм, основанный на числах Фибоначчи, приводит к цели за 5 шагов. Докажите, что при неблагоприятной ситуации знание возраста 4 деревьев недостаточно.

14. У скольких деревьев необходимо определить возраст (сколько нужно шагов) по бинарному алгоритму для $n = 2^m$?

15 ([6], М.Островский). Загадано число от 1 до 144. Разрешается выделить одно подмножество множества чисел от 1 до 144 и спросить, принадлежит ли ему загаданное число. За ответ «да» надо заплатить 2 рубля, за ответ «нет» – 1 рубль. Какая наименьшая сумма денег необходима для того, чтобы наверняка угадать число? (Само число нужно назвать, т.е. 2 рубля в любом случае придется заплатить.)

Задачи для исследования

Мы рассматривали алгоритмы, связанные с числовыми последовательностями. Однако на современных компьютерах устанавливают и графические редакторы. В связи с этим предлагаем вам новые задачи.

Задача 7. Задача о графическом редакторе

Графический редактор на плоскости может выполнять операции копирования следующего вида: 1) скопировать и одновременно отразить фигуру относительно какой-то выбранной оси. 2) скопировать текущую фигуру или фигуру из памяти и параллельно ее сдвинуть. При таких операциях старая фигура остается на месте. Полученные фигуры на плоскости можно сохранить в памяти компьютера. За какое минимальное число операций копирования из исходной фигуры можно получить заданную, если:

а) исходная фигура – квадрат 1×1, заданная фигура – квадрат $n\times n$;

б) исходная фигура – квадрат 1×1, заданная фигура – прямоугольник $n\times m$?

в) исходная фигура – равносторонний треугольник со стороной 1, заданная фигура – треугольник с длиной стороны n;

г) Исследуйте случай, когда исходная фигура – правильный шестиугольник со стороной 1.

Например, из квадрата 1×1 квадрат 2×2 можно получить за две операции так: сначала с помощью симметрии или параллельного переноса получаем доминошку 1×2, затем, аналогично, либо параллельным переносом, либо симметрией относительно длинной стороны получаем квадрат 2×2.

Попробуйте исследовать случай, когда разрешена только операция 1), а операция 2) недоступна.

В случае, когда исходная фигура является квадратом 1×1, мы можем применить результаты, полученные для задач 1–4.

Упражнение 16. Решите задачи 7 а), б) для исходной фигуры – квадрата 1×1.

Случаи треугольника и шестиугольника являются более сложными, и мы оставляем их для исследования.

В следующей задаче мы предлагаем заменить удвоение умножением на какое-то заданное число p. Какой алгоритм будет быстрее: p-арный, метод множителей или метод дерева степеней? Ответ на эту задачу авторам неизвестен.

Задача 8. а) У Пети и Васи есть калькуляторы с двумя операциями. Петин калькулятор умеет:

- умножить число на натуральное число p;
- увеличить число на 1.

Васин калькулятор умеет:

- умножить число на натуральное число q;
- увеличить число на 1.

Известно, что p и q взаимно простые числа, при этом $1 < p < q$. В начальный момент на обоих калькуляторах нули. Можно ли указать все натуральные числа, которые Петя может посчитать за меньшее число операций, чем Вася?

б) Исследуйте задачу для $p = 2$, $q = 3$.

Упражнение 17. Докажите, что существует бесконечно много чисел, которые Васин калькулятор может подсчитать за меньшее число операций, чем Петин калькулятор. Приведите примеры чисел, которые Петин калькулятор может подсчитать за меньшее число операций, чем Васин калькулятор.

Литература

1. *Е.А.Морозова, И.С.Петраков, И.Ф.Скворцов*. Международные математические олимпиады. – М.: Просвещение, 1976. – Задача 105.
2. *E. de Jonquiéres.* Question 49 (H.Dellac). – L'Intermédiaire Math. V.1(20), 1894. – P.162–164.
3. Геометрические олимпиады имени И.Ф.Шарыгина. Составители А.А.Заславский, В.Ю.Протасов, Д.И.Шарыгин. – М.: МЦНМО, 2007. – С.124.
4. *Н.Х.Агаханов, И.И.Богданов, П.А.Кожевников и др.* Математика. Областные олимпиады. – М.: Просвещение, 2010. – Задача 258.
5. *Н.Х.Агаханов, И.И.Богданов, П.А.Кожевников и др*. Математика. Всероссийские олимпиады. – М.: Просвещение, 2008. – Задачи 9.3; 9.6.
6. *Н.Х.Агаханов, И.И.Богданов, П.А.Кожевников и др.* Всероссийские олимпиады школьников по математике 1993–2009. Заключительные этапы. – М.: МЦНМО, 2010. – Задачи 344; 160.
7. *Д.Э.Кнут*. Искусство программирования. Т.2, разд.4.6.3.
8. *N.M.Clift*. Calculating optimal addition chains. – Computing. V.91, 2011. – P.265–284.
9. *С.Б.Гашков.* Системы счисления и их применение. – М.: МЦНМО (Библиотека «Математическое просвещение», вып.29), 2004.
10. *С.Б.Гашков.* Задача об аддитивных цепочках и ее обобщения. – «Математическое просвещение», третья серия, т.3, вып.15, 2011.