Косовская Татьяна Матвеевна,
Косовский Николай Кириллович

ОСНОВЫ ДОКАЗАТЕЛЬСТВ ПОЛИНОМИАЛЬНОЙ БЫСТРОТЫ ПРОСТЕЙШИХ МАТЕМАТИЧЕСКИХ АЛГОРИТМОВ

Аннотация

Статья посвящена проблемам доказательства полиномиальной вычислимости функций, другими словами, доказательства принадлежности функций классу **FP**. Доказательства могут производиться с помощью паскалевидных функций, которые более удобны программистам, чем модели вычислений, основанные на машине Тьюринга. Будет приведена идея доказательства теоремы, что если не только число шагов вычисления паскалевидной функции, но и размер записи всех промежуточных вычислений не превосходят полинома от длины записи исходных данных, то эта функция принадлежит классу **FP** и обратно.

Ключевые слова: P, **FP**, **NP**, машина тьюринга, паслкалевидные функции, NP-полнота, NP-трудность, примеры NP-полных задач.

ВВЕДЕНИЕ ИЛИ ИСТОРИЯ ВОЗНИКНОВЕНИЯ РАССМАТРИВАЕМОЙ ТЕМАТИКИ

Приводятся удобные для программистов доказательства и математически обоснованные сомнения полиномиальной быстроты простейших математических алгоритмов.

Развитие теории алгоритмов началось в связи с необходимостью доказывать невозможность построения алгоритма, решающего ту или иную массовую задачу. С этой целью были введены в рассмотрение первые математические понятия алгоритма: рекурсивные функции и машины Тьюринга (см, например, [1–3]). Был сформулирован тезис Черча (а впоследствии и тезис Тьюринга-Черча), утверждавший, что «всякая интуитивно вычислимая функция является рекурсивной» (соответственно, «всякая интуитивно вычислимая функция может быть вычислена на машине Тьюринга»). Этот тезис нельзя ни доказать, ни опровергнуть, так как понятие «интуитивно вычислимая функция» каждый, вообще говоря, может понимать по-своему.

В наше время существует большое количество математических понятий алгоритмов, для большинства из которых доказано следующее подтверждение тезиса Черча (тезиса Тьюринга-Черча): «Всякая функция, вычислимая с помощью одного математического понятия алгоритма, вычислима и с помощью другого».

С развитием вычислительной техники стало ясно, что существование алгоритма, решающего задачу, недостаточно для

его использования с целью многократного применения к различным исходным данным. На первый план вышли время работы алгоритма и объем используемой памяти (см., например, [1–6, 8]), называемые соответственно временно́й и емкостной сложностью алгоритма.

Наиболее распространенной в настоящее время классификацией сложности алгоритмов является классификация, основанная на ограничении числа шагов или памяти функциями определенного вида (см., например, [1–5]). Приведем пример определения трех наиболее распространенных таких классов – классов **P**, **FP**, **NP**.

Определение. *Предикат принадлежит классу* **P***, если существует машина Тьюринга, решающая эту задачу, число шагов которой не превосходит полинома от длины записи ее исходных данных.*

Если в этом определении слово «предикат» заменить на слово «функция», то получим определение класса **FP**. Если же в нем заменить слова «машина Тьюринга» на «недетерминированная машина Тьюринга», то получим определение класса **NP**.

К началу XXI века большинство математиков, занимающихся оценками сложности вычислений у алгоритмов, предназначенных для использования на компьютерах, стали понимать важность изучения их принадлежности классу **FP**.

Более того, считается, что класс **FP** совпадает с классом эффективных вычислений. Такое интуитивное понятие как «эффективное вычисление» было формализовано с помощью расширенного тезиса Черча (см., например, [8]): «Функция, вычислимая за полиномиальное время на разумной вычислительной модели, использующей разумное измерение временной сложности, вычислима на детерминированной машине Тьюринга за полиномиальное время».

Представление класса **FP** невозможно с помощью ограниченного произведения, обозначаемого посредством (П), которое может вывести за пределы этого класса. За его пределы не выводят суперпозиция и ограниченная возвратная словарная рекурсия совместно определяемых функций (совместная ограниченная возвратная словарная рекурсия).

Существенная трудность доказательства принадлежности алгоритма классу **FP** связана с тем, что многие численные алгоритмы используют косвенную адресацию при обработке массивов, для которой были созданы модели вычисления РАМ и РАСП – равнодоступная адресная машина (машина с произвольным доступом к памяти) и она же с хранимой программой (машина с произвольным доступом к памяти и хранимой программой) [1].

Ниже оценку числа шагов предлагается использовать в традиционном смысле, в частности, предложенном в [1]. Кроме того, можно использовать и БульРАСП (адресную машину с хранимой памятью, предложенную переводчиком книги [1] в конце раздела 1.4). То есть один шаг учитывается вне зависимости от длины записи операндов команды.

К сожалению, в этом случае при наличии операции умножения за полиномиальное число шагов на этих моделях можно вычислить функцию экспоненциального вида, например x^y (при быстром вычислении), что невозможно сделать за полиномиальное число шагов на машине Тьюринга (длина записи результата задается экспонентой от длины записи y).

Ниже будет предложено математическое понятие целочисленного алгоритма, основанное на использовании широко распространенного языка программирования Паскаль. Это понятие является обобщением РМА – математической модели вычисления компьютерными алгоритмами. Будет доказано, что если не только число шагов вычисления функции, но и размер записи всех промежуточных вычислений не превосходят полинома от длины записи исходных данных, то эта функция принадлежит классу **FP**, и обратно.

1. МАССОВЫЕ ЗАДАЧИ

Среди задач дискретного математического характера выделяются индивидуальные задачи, не содержащие аргументов и

параметров. Для таких задач достаточно один раз ее решить и знать ответ. В большинстве случаев формулировка задачи содержит аргументы, которые принято называть исходными данными решающего ее алгоритма.

Определение. *Массовой задачей называется задача, в которой имеются переменные-аргументы, которые могут принимать значения из* ***бесконечного*** *множества дискретных математических конструктивных объектов, например множества целых чисел, множества рациональных чисел, множества слов в заранее заданном конечном алфавите.*

В общем виде массовую задачу можно записать в виде $(?\,\bar{x})\ \varphi(\bar{x})$, где $\varphi(\bar{x})$ – формула какого-либо формализованного языка со списком переменных (аргументов задачи) $\bar{x}$. Это выражение читается так: «При каких значениях параметров списка $\bar{x}$ верна формула $\varphi(\bar{x})$?». Простейшими примерами массовых задач могут служить, например, следующие.

1. $(?\,abc)\exists x\,(x \in \mathbf{Z}\ \&\ ax^2 + bx + c = 0$ – при каких целых значениях параметров a, b, c уравнение $ax^2 + bx + c = 0$ имеет целое решение?

2. $(?\,abc)\forall x\,(x \in [0,\ 1] \cap \mathbf{Q} \Rightarrow ax^2 + bx + c > 0)$ – при каких рациональных значениях параметров a, b, c квадратный трехчлен $ax^2 + bx + c$ положителен на отрезке [0, 1]?

3. $(?P)P$ – какие пропозициональные формулы тождественно истинны?

4. $(?Q)\,\tilde{\forall}\,Q$ – какие предикатные формулы общезначимы?

В последней задаче знак $\tilde{\forall}$ означает кванторы всеобщности по всем предметным переменным, свободно входящим в Q.

Под серией массовых задач будем понимать массовую задачу, которая содержит один или несколько дополнительных параметров, которые также могут принимать значения из бесконечного множества конструктивных объектов. Такие дополнительные параметры будем называть параметрами серии.

Алгоритмы, решающие массовые задачи, в зависимости от длины записи исходных данных, могут совершать различное число шагов (или макро-шагов – этапов, объединяющих, как правило, полиномиальное число стандартных шагов) в используемой математической модели вычисления.

Для каждой математической модели алгоритмов M класс алгоритмов, число шагов которых не превосходит полинома от длины записи исходных данных, будем обозначать посредством $\mathbf{FP}_M$. Если при этом дополнительно результат работы алгоритма принимает значение только из двухэлементного множества, то такой алгоритм называют предикатом, а множество всех таких предикатов принято обозначать посредством $\mathbf{P}_M$. Таким образом, $\mathbf{P}_M \subsetneqq \mathbf{FP}_M$.

Следует отметить, что так определенные классы $\mathbf{P}_M$ и $\mathbf{FP}_M$ для разных математических моделей алгоритма окажутся различными. Так, например, если в качестве такой модели взяты только аддитивные операции, то функцию 2^{2^n} невозможно вычислить за полином от длины записи n. В то же время если в математической модели вычисления разрешено использование операции умножения двух чисел (или вычисления квадрата числа), то 2^{2^n} вычислима за полином от длины записи n шагов. При дополнительной возможности использования операции возведения в степень она вычисляется за два шага.

Важной задачей математической теории информатики является поиск массовых задач, решаемых алгоритмами из класса **FP**. При этом возможна ситуация, когда каждая массовая задача некоторой серии может быть решена алгоритмом из **FP**. И в то же время превращение этой серии в новую массовую задачу (путем превращения всех параметров серии в аргументы новой массовой задачи, которые, как правило, называют исходными данными задачи) приводит к тому, что до сих пор не известен (и, по мнению большинства математиков, вряд ли будет известен) алгоритм из **FP**, решающий новую массовую задачу.

Важным классом задач являются NP-полные задачи [1–6] – самые долгорешаемые задачи из класса **NP**. В настоящее время накоплено более тысячи массовых задач, относительно которых доказано, что они являются NP-полными (ранее, более 20 лет назад, для NP-полных задач в русском языке использовался термин «переборная задача»). Все эти задачи являются кандидатами на невозможность решения их ни одним алгоритмом из класса **FP**. Более того, доказательство принадлежности алгоритма, решающего такую задачу, классу **FP** или доказательство не принадлежности ему в 2000 году было оценено американским математическим сообществом в один миллион долларов.

Массовые задачи предназначены для непосредственного решения их на компьютерах. В дальнейшем слово «массовая» будет, как правило, опускаться для краткости.

2. ПАСКАЛЕВИДНЫЕ ФУНКЦИИ КАК МАТЕМАТИЧЕСКОЕ ПОНЯТИЕ АЛГОРИТМА

Как мы уже говорили, полиномиальное число шагов для разных математических понятий алгоритмов не обязательно обеспечивает его принадлежность классу **FP**. В то же время, программирование на машине Тьюринга не вполне адекватно современной практике программирования. В связи с этим введем математическое понятие целочисленного алгоритма на основе широко распространенного языка Паскаль. Более точно, в основе рассматриваемого понятия лежит предложенный Н. Виртом язык Паскаль [7] с включением динамических массивов (то есть массивов с динамически определяемыми верхними границами), а не другие его различные реализованные версии.

Пусть в языке Паскаль переменные типа *integer* могут принимать в качестве значения любое целое число (то есть целое число с любой сколь угодно большой длиной записи), а другие основные типы переменных не используются. Как иногда говорят, возможно использование только констант и целых чисел произвольной разрядности.

Определение. *Целочисленным алгоритмом (паскалевидной функцией) назовем функцию предложенного Н. Виртом языка Паскаль, обрабатывающую ее фактические параметры – целые числа, поступающие на ее вход, и не использующую файлы, записи, множества и указатели, а также процедуры и функции, имеющие в качестве параметров свои процедуры и функции.*

Основой при получении оценок сложности вычислений является определение того, что же считать шагом вычисления и что считать длиной используемой памяти. С определением шага вычисления, как правило, трудностей не возникает. Но мы уже видели, что на разных моделях вычисления функция 2^{2^n} вычисляется за очень разное число шагов.

Определение. *Вычисление каждого оператора языка Паскаль из определения функции (включая оператор цикла и оператор присваивания выражения) считается за один шаг. Каждое обращение к ранее определенной функции или процедуре (то есть ее вызов) также считается за один шаг. К каждому шагу прибавляется число всех дополнительных шагов, необходимых для его выполнения, включая и все рекурсивные вызовы. (Такое обращение возможно и при вычислении арифметического или булевого выражения.)*

На каждом шаге вычисления функции будем дополнительно вычислять длину записи вычисления на этом шаге.

Пусть t – арифметическое или булево выражение. Процесс его вычисления может быть представлен в виде корневого дерева. Глубиной записи t назовем глубину этого дерева.

Определение. *Длиной записи вычисления выражения t на глубине i назовем сумму длин абсолютных величин всех подвыражений глубины i выражения t.*

Длиной записи вычисления выражения t назовем максимум по всем i (от 1 до глу-

бины t) длин записи вычисления t на глубине i.

Определение. *На начальном шаге длина записи вычисления совпадает с длиной записи исходных данных (значений для аргументов функции). На заключительном шаге вычисления функции длина записи вычисления совпадает с длиной записи абсолютной величины результата вычисления вызываемой функции.*

Под длиной записи промежуточных вычислений (то есть используемой памяти) понимаем максимум длины записей вычисления по всем шагам этого вычисления, включая начальный и заключительный.

Для удобства читателя предполагаем, что все числа записаны в десятичной системе счисления (хоть основание системы счисления, отличное от единицы, совершенно не принципиально).

3. ПОЛИНОМИАЛЬНО БЫСТРЫЕ ВЫЧИСЛЕНИЯ

Поскольку ниже будет идти речь о полиномиально быстрых вычислениях (то есть о классе **FP**), то требуется доказывать, что длина записи промежуточных вычислений и число шагов вычисления не превосходят полинома от длины записи исходных данных.

Определение. *Целочисленный алгоритм (паскалевидная функция) назовем дважды полиномиальным, если как число его шагов, так и длина записи промежуточных вычислений не превосходят полинома от длины записи исходных данных.*

Отметим, что паскалевидные функции для обработки целых чисел могут использовать целые числа сколь угодно большой длины записи. В связи с этим острая необходимость использования нединамических массивов отпадает. Вместо массивов можно использовать дважды полиномиальные функции c, l и r и их итерации, где функция c нумерует пары. Она обладает свойством

$$c(x, y) = c(x',y') \Rightarrow x = x' \ \& \ y = y',$$
$$l(c(x, y)) = x, \ r(c(x, y)) = y$$

Примером такой функции с может служить $c(x, y)$ (см. рис. 1).

Построение функций l и r оставляется читателю в качестве упражнения по программированию на паскале. Поскольку функция $(x + y)(x + y + 1)/2 + x$ нумерует натуральными числами (включая ноль) множество всех пар натуральных чисел (в том числе и содержащие ноль), то предложенная функция c является отображением всех пар целых чисел на множество всех целых чисел.

Значения всех элементов массива $x[1..n]$ могут быть представлены в виде числа $c(n, \underbrace{c(...c(}_{n-1 \text{ раз}} x[1], x[2], ... x[n])...)...)$.

С помощью итераций функций l и r можно выразить любой элемент массива x.

Отметим, что изобретатель языка Паскаль не включил в него операцию возведения в степень, использование которой, как правило, не является полиномиально быстрым. Впрочем, он включил в него операцию SQR (то есть функцию возведения в квадрат), которая при использовании в выражении n раз позволяет вычислить 2^{2^n} (см. в приложении также теорему 1).

В приложении будет изложена идея доказательства следующей теоремы.

Теорема. *Класс дважды полиномиальных паскалевидных функций совпадает с классом* **FP**.

$$c(x,y)=\begin{cases}2((x+y)(x+y+1)/2+x, & \text{если } x,y\ge 0\\ 2((x+y)(x+y+1)/2+x-1, & \text{если } x,y\le 0 \text{ и } (x,y)\ne(0,0)\\ -2((x+y)(x+y+1)/2+x, & \text{если } x>0 \text{ и } y<0\\ -22((x+y)(x+y+1)/2+x-1, & \text{если } x<0 \text{ и } y>0\end{cases}$$

Рис. 1

4. NP-ПРЕДИКАТЫ

Дадим новое определение недетерминированных полиномиальных предикатов, которое будет базироваться на паскалевидных функциях.

Определение.

Предикат Q будем называть NP-предикатом, если имеется дважды полиномиальная паскалевидная функция, которая по каждому набору исходных данных для нее и дополнительному аргументу для целых чисел, не превосходящих по длине записи некоторого полинома от длины записи исходных данных, и равная 1 тогда и только тогда, когда истинен предикат Q.

Говоря более формализовано, NP-предикат Q получается из дважды полиномиальной паскалеовидной функции f по следующей формуле

$$Q(\bar{x}) \Leftrightarrow \exists y \left(y \le p(\| \bar{x} \|) \,\&\, f(\bar{x}, y) = 1 \right),$$

где $p(\| \bar{x} \|)$ – некоторый полином от длины записи списка аргументов $\bar{x}$.

Множество всех NP-предикатов образует класс **NP**.

Ранее уже упоминалась проблема, оцененная в один миллион долларов. Более кратко она может быть сформулирована в виде: «доказать или опровергнуть, что **P = NP**».

Определенные здесь классы **P** и **NP** – те же самые, что и в традиционном смысле (приведенные ранее). Доказательство этого основано на теореме 4 из приложения.

5. ИСХОДНЫЕ ПРИМЕРЫ NP-ПОЛНЫХ ПРЕДИКАТОВ

Приведем первый пример NP-полного предиката (см., например, [1, 2]). Его принято называть массовой задачей ВЫПОЛНИМОСТЬ (сокращенно ВЫП или SAT). Он представляет собой проверку выполнимости пропозициональной (или, как часто говорят, булевой) формулы, находящейся в конъюнктивной нормальной форме, то есть представляющую собой многократную конъюнкцию элементарных дизъюнкций (многократных дизъюнкций переменных или их отрицаний). Под выполнимостью понимается возможность подбора таких булевых констант, подставляемых вместо переменных, которые обеспечивают истинное значение заданной на входе пропозициональной формулы.

Будем говорить, что *предикат* Q_1 *полиномиально m-сводится к предикату* Q_2 и писать $Q_1 \le_m^p Q_2$, если можно построить дважды полиномиальную функцию f такую, что $\forall x \left(Q_1(\bar{x}) \Leftrightarrow Q_2\left(f(\bar{x})\right)\right)$.

Легко доказать, что введенное отношение является отношением порядка, то есть оно рефлексивно и транзитивно.

Определение. *NP-предикат Q назовем NP-полным, если ВЫП* $\le_m^p$ *Q.*

Это определение отличается от общепринятого [1, 2], но эквивалентно ему и более удобно для применения, а по существу именно оно (и транзитивность полиномиальной m-сводимости) используется в практике доказательства NP-полноты всех задач, кроме задачи ВЫПОЛНИМОСТЬ.

Задача 3-ВЫПОЛНИМОСТЬ (сокращенно 3-ВЫП или 3-SAT) является сужением задачи выполнимость, при котором каждая элементарная дизъюнкция формулы содержит ровно три переменные.

Теорема. ВЫП $\le_m^p$ 3-ВЫП.

Доказательство. Легко видеть, что выполнимость дизъюнкции $y_1 \vee \ldots \vee y_k$, где $k \ge 3$, равносильна выполнимости $((y_1 \vee y_2) \Leftrightarrow z_1) \,\&\, ((y_3 \vee z_1) \Leftrightarrow z_2) \,\&\, \ldots \,\&\, ((y_n \vee z_{n-2}) \Leftrightarrow z_{n-1}) \,\&\, z_{n-1}$, где z_1, ..., z_{n-1} – новые переменные. Формула вида $(w_1 \vee w_2) \Leftrightarrow w_3$ эквивалентна по выполнимости формуле

$$(((w_1 \vee w_2) \Rightarrow w_3) \,\&\, (w_3 \Rightarrow (w_1 \vee w_2)),$$

которая, в свою очередь, эквивалентна $(\neg w_1 \vee w_3) \,\&\, (\neg w_2 \vee w_3) \,\&\, (w_1 \vee w_2 \vee \neg w_3)$. Осталось заменить формулу вида $w_1 \vee w_2$ на равносильную ей по выполнимости

$$(w_1 \vee w_2 \vee w_3) \,\&\, (w_1 \vee w_2 \vee \neg w_3),$$

где w_3 – новая переменная, а формулу вида w_1 на $(w_1 \vee w_2 \vee w_3)$ & $(w_1 \vee w_2 \vee \neg w_3)$ & $(w_1 \vee \neg w_2 \vee w_3)$ & $(w_1 \vee \neg w_2 \vee \neg w_3)$, где w_2, w_3 – новые переменные.

Можно оценить сверху длину вновь построенной формулы посредством $C \cdot n \cdot \log n + C_0$ (при некоторых положительных C и C_0), где n – длина исходной формулы при некоторых положительных C и C_0. Логарифмический сомножитель появляется из-за необходимости введения новых переменных, которые можно записывать в алфавите из нескольких букв. (В случае необходимости использовать только однобуквенный алфавит верхняя оценка увеличится до $Cn^2 + C_0$ при некоторых положительных C и C_0). ▪

Примером NP-полной задачи из другой области математики может служить задача булева линейного программирования, то есть задача проверки совместности в числах из {0, 1} линейных целочисленных нестрогих неравенств.

Теорема. *Задача 3-ВЫП полиномиально m-сводится к задаче булева линейного программирования.*

Доказательство. Выполнимость каждой элементарной дизъюнкции вида $x \vee \neg y \vee z$ эквивалентна совместности в целых числах системы нестрогих неравенств

$$x \geq 0,\ y \geq 0,\ z \geq 0,\ x \leq 1,\ y \leq 1,\ z \leq 1,$$
$$x + (1 - y) + z \geq 1.$$

Верхняя оценка длины записи системы линейных неравенств, соответствующих формуле из условия задачи 3-ВЫП, длина записи которой равна n, может иметь вид $Cn + C_0$ при некоторых положительных C и C_0.

Полученная система линейных целочисленных нестрогих неравенств будет иметь решение в целых числах тогда и только тогда, когда она будет иметь решение в числах из {0, 1} в силу неравенств для переменных. При этом значение 1 соответствует истине, а значение 0 – лжи. ▪

Анализируя только что приведенное доказательство получаем еще две NP-полные задачи: задача проверки совместности в целых числах (и в числах из {0, 1}) системы линейных целочисленных нестрогих неравенств, каждое неравенство в которой содержит не более трех переменных.

Заменяя систему неравенств на систему уравнений, можно задачу 3-ВЫП полиномиально m-свести к задаче проверки совместности в числах из {0, 1} системы целочисленных линейных уравнений. Действительно, выполнимость каждой элементарной дизъюнкции вида $x \vee \neg y \vee z$ эквивалентна совместности в числах из {0, 1} уравнения $x + (1 - y) + z = 1 + u + v$, где u и v – новые переменные. Верхняя оценка длины записи новой системы уравнений при кодировании переменных словами в многобуквенном алфавите может быть оценена посредством $C \cdot n \cdot \log n + C_0$ при некоторых положительных C и C_0.

Итак, NP-полна задача проверки совместности в числах из {0, 1} системы целочисленных линейных уравнений, каждое из которых содержит ровно 5 переменных.

Аналогично можно доказать, что NP-полна задача проверки совместности в числах из {0, 1} системы целочисленных линейных дизуравнений, каждое из которых содержит ровно три переменные. Доказательство основывается на эквивалентностях, аналогичных $x \vee \neg y \vee z \Leftrightarrow x + (1 - y) + z \neq 0$.

При программировании на языке Паскаль дважды полиномиальных паскалевидных функций, осуществляющих описанные здесь сведéния, следует иметь в виду, что каждая система неравенств (и уравнений) записывается словом в небольшом конечном алфавите, буквы которого могут рассматриваться как ненулевые цифры числа, представляющего это число в некоторой системе счисления при достаточно большом основании. Само программирование оставляется читателю в качестве длительно выполняемого, но простого задания на программирование средствами языка Паскаль в рамках дважды полиномиальных паскалеобразных функций.

Без доказательства (из-за его большой длины, в чем можно убедиться, посмотрев, например, [2]), приведем еще одну NP-полную задачу, известную под названием СУММА РАЗМЕРОВ. Она представляет собой проверку разрешимости в числах из {0, 1} линейных уравнений вида $a_1x_1 + ... + a_nx_n = b$, где $a_1,... a_n$, b – положительные целые числа.

Следует отметить, что если задача имеет числовые параметры, то важно, записаны ли значения этих параметров в позиционной системе счисления с основанием, отличным от единицы, или в так называемой унарной системе, то есть число a имеет вид $\underbrace{1...1}_{a}$.

Назовем задачей СУММА УНАРНЫХ РАЗМЕРОВ задачу, аналогичную задаче СУММА РАЗМЕРОВ, в которой все числа a_1, ..., a_n, b записаны в унарной системе счисления.

Можно доказать, что используя метод динамического программирования, новую задачу всегда можно решить с использованием дважды полиномиальной паскалеобразной функции. Идею ее реализации можно посмотреть, например, в [2].

ПРИЛОЖЕНИЕ

Пусть N_2 – множество двоичных записей натуральных чисел (включающее число 0), знак ^{2} используется для обозначения функции возведения в квадрат, $\overline{\&}$ и $\overline{\vee}$ – поразрядные конъюнкция и дизъюнкция соответственно.

Заметим, что если задача проверки истинности предиката вида $\exists Y\, P(X, Y)$ является NP-полной, то задача нахождения того значения Y, для которого эта формула верна (иногда пишут $(?Y)\ P(X, Y)$) является NP-трудной. Точное определение NP-трудной задачи имеется, например, в [2, 8].

Терм в заданной сигнатуре представляет собой формализованную запись арифметического выражения, в которой использованы константы, переменные и функции из этой сигнатуры. Постоянный терм не содержит переменных.

Теорема 1. *Задача вычисления знака постоянного терма в сигнатуре* $\left\langle N_2;+, -, \cdot, /, {}^2, \overline{\&}\right\rangle$ *является NP-трудной.*

Доказательство. Сведем задачу ВЫПОЛНИМОСТЬ (ВЫП) к предикатной модификации, рассматриваемой в формулировке теоремы задачи: «проверка положительности постоянного терма в сигнатуре $\left\langle N_2;+, -, \cdot, /, {}^2, \overline{\&}\right\rangle$».

Пусть исходными данными для задачи ВЫП являются пропозициональные переменные x_1, ..., x_r и элементарные дизъюнкции c_1, ..., c_m. Воспользуемся тем, что $2^r = \underbrace{2\cdot ...\cdot 2}_{r\text{ раз}}$. Каждой предметной переменной x_i ($i = 1, ..., r$) ставим в соответствие число z_i, кодирующее i-ый столбец в списке всех наборов значений переменных x_1, ..., x_r. Точнее, $z_r = \overline{1^{\langle r\rangle}0^{\langle r\rangle}}$, а при $i \in [1, r-1]$

$$z_i = \overline{1^{\langle p\rangle}0^{\langle p\rangle}} + 2^{2p}\cdot\overline{1^{\langle p\rangle}0^{\langle p\rangle}} + ... + 2^{2pq}\cdot\overline{1^{\langle p\rangle}0^{\langle p\rangle}} =$$

$$= \left(2^p - 1\right)\cdot 2^p\cdot\left(1 + 2^{2p} + ... + 2^{2pq}\right) = \left(2^p - 1\right)\cdot 2^p\cdot\frac{2^{2p(q+1)} - 1}{2^{2p} - 1},$$

где $p = 2^{i-1}$, $q = 2^{r-i} - 1$, $b^{\langle k\rangle}$ – слово, состоящее из букв b, повторенных k раз, $\overline{S}$ – двоичное число, соответствующее слову S, состоящему из двоичных цифр.

Отметим, что экспоненты, появившиеся в формуле, моделируются постоянными термами в заданной сигнатуре. А именно

$$2^p = 2^{i-1} = (...(2)\underbrace{^2...)^2}_{i-1\text{ раз}},$$

$$2^{2p} = 2^{2\cdot 2^{i-1}} = 2^{2^i} = (...(2)^{\underbrace{2}_{i\text{ раз}}}...)^2,$$

$$2^{2p(q+1)} = 2^{2\cdot 2^{i-1}\cdot 2^{r-i}} = 2^{2^r} = (...(2)^{\underbrace{2}_{r\text{ раз}}}...)^2.$$

Таким образом, каждой переменной x_i ($i = 1, ..., r$) не более чем за полином шагов от длины записи исходных данных для ВЫП ставится в соответствие постоянный терм, содержащий функцию возведения в квадрат.

Операцией r-поразрядного отрицания $\neg$ константы из N_2, длина записи которой не превосходит r, будем называть замену всех символов 1 в записи на символ 0, а всех символов 0 (включая ведущие нули, отсутствующие в записи длины r) на символ 1. В арифметической записи $\neg a = 2^r - a - 1$, если $|a| \le r$.

Поразрядная дизъюнкция может быть выражена через r-поразрядное отрицание и поразрядную конъюнкцию по обычным формулам, при этом длина записи терма при исключении поразрядной дизъюнкции вырастет линейно.

Истинность всех дизъюнкций $c_1, ..., c_m$ хоть на одном наборе значений переменных равносильна тому, что значение постоянного терма, полученного из пропозициональной формулы $c_1\&...\&c_m$ заменой & на $\overline{\&}$, $\vee$ на $\overline{\vee}$, $\neg$ на $\neg$ и переменных $x_1, ..., x_r$ на построенные постоянные термы, больше нуля. Таким образом, задача ВЫП полиномиально сводится к проверке положительности постоянного терма в сигнатуре $\left\langle N_2;+,-,\cdot,/,{}^2,\overline{\&}\right\rangle$. ■

Несмотря на NP-трудность задачи из формулировки теоремы 1, доказанной в [6], можно доказать полиномиальность задачи вычисления постоянных термов при ограничении на его длину записи. Пусть $\|t\|$ – длина записи терма t, $[t]$ – значение постоянного терма t.

Теорема 2. *Каково бы ни было множество постоянных термов в конечной сигнатуре, содержащей функции только из* **FP**, *задача вычисления постоянного терма из этого множества принадлежит* **FP**, *если при некоторой константе E для всякого подтерма t вычисляемых термов выполняется* $\|[t]\| \le \|t\|^E + E$.

Доказательство.

Во все функции рассматриваемой сигнатуры вводим дополнительный фиктивный аргумент, вместо которого подставляем значение $\|t\|^E$, для которого выполняется неравенство из условия теоремы. При этом длина записи терма $\|\{t\}\|$ увеличится не более, чем полиномиально.

Все функции становятся неудлиняющими, то есть длина записи результата каждой из них не превосходит суммы длин записей ее аргументов (включая длину записи фиктивного аргумента). Следовательно, можно воспользоваться следующей теоремой из [6]. ■

Теорема 3. *Пусть имеется конечный список функций $f_1, ..., f_m$, вычислимых на детерминированной машине Тьюринга за полиномиальное время, причем время вычисления каждой функции f_j ($j = 1, ..., m$) не превосходит $C\cdot s_j^l$ (где s_j – сумма длин записей аргументов, C, l – некоторые константы). И пусть длина записи значения каждой функции не превосходит суммы длин записей ее аргументов.*

Тогда вычисление постоянного терма в сигнатуре $\left\langle N_2; f_1, ..., f_m\right\rangle$ осуществимо на детерминированной машине Тьюринга за полиномиальное время.

Точнее, время вычисления этого терма не превосходит $C\cdot L\cdot n^l \le C\cdot n^{l+1}$, где L – глубина терма, n – длина записи терма, C – константа.

Доказательство проведем методом возвратной математической индукции по глубине терма L.

При $L = 0$ (то есть терм является константой) теорема очевидна.

Рассмотрим терм $t = f(t_1, ..., t_k)$ глубины L, где $t_1, ..., t_k$ – термы глубин $L_1, ..., L_k$ соответственно ($L_i < L$). По индукционному предположению t_i вычисляется не более чем за $C\cdot L_i\cdot n_i^l$ шагов, где длина записи t_i равна n_i ($i = 1, ..., k$).

Пусть $|S|$ – длина записи слова S, $[\,t\,]$ – значение постоянного терма t.

По условию теоремы длина записи значения терма t не превосходит суммы длин записей термов t_1, ..., t_k. Используя это, дополнительной возвратной математической индукцией по L можно доказать, что $|[t]| \le \sum_{i=1}^{k} |[t_i]| \le \sum_{i=1}^{k} n_i < n$. По индукционному предположению время вычисления терма t_i не превосходит $C \cdot L_i \cdot n_i^{\,l}$.

По условию теоремы время вычисления $f([t_1], ...,[t_k])$ не превосходит $C \cdot \left(\sum_{i=1}^{k} |[t_i]|\right)^l \le C \cdot n^l$, а общее время вычисления терма t не превосходит

$$C \cdot n^l + \sum_{i=1}^{k} C \cdot L_i \cdot n_i^l \le$$

$$C \cdot n^l + C \cdot (L-1) \cdot \sum_{i=1}^{k} n_i^l \le$$

$$C \cdot n^l + C \cdot (L-1) \cdot \left(\sum_{i=1}^{k} n_i \right)^l =$$

$$C \cdot n^l + C \cdot (L-1) \cdot n^l = C \cdot L \cdot n^l . \quad \blacksquare$$

Следствие теоремы 2. *Какова бы ни была конечная сигнатура функций из* **FP**, *вычисление постоянных термов в этой сигнатуре принадлежит классу* **FP**, *если для некоторой константы C верно* $\forall \bar{x} \left(\| f(\bar{x}) \| \le \sum_i \| x_i \| + C \right)$.

Это следствие усиливает теорему 3.

Теорема 4. *Класс дважды полиномиальных паскалевидных функций совпадает с классом* **FP**.

Идея доказательства. Каждая конфигурация машины Тьюринга МТ (то есть непустое содержимое ленты вместе с указанием положения головки и номера состояния машины, ограниченное слева и справа бланковыми символами) может быть промоделирована двумя динамическими массивами, первые элементы которых кодируют бланковый символ ленты.

Длина записи числа в этом массиве будет ограничена полиномом от длины записи исходных данных, если МТ совершает число шагов, ограниченное полиномом от длины записи исходных данных. Таким образом, каждая функция из класса **FP** может быть промоделирована дважды полиномиальной паскалевидной функцией.

Докажем, что всякая дважды полиномиальная паскалевидная функция принадлежит классу **FP**. Элементы массивов располагаем на дополнительной ленте машины Тьюринга в виде последовательности пар вида ($[\,\bar{i}\,]$, $[a(\bar{i})]$), где $[\,\bar{i}\,]$ – список значений индексов, $[a(\bar{i})]$ – значение элемента массива.

Во-первых, отметим, что каждая операция языка Паскаль принадлежит классу **FP**. Более того, ее можно сделать неудлиняющей добавлением фиктивного аргумента, ограничивающего сверху длину записи ее результата. Длина записи этого результата не превосходит полинома от длины записи первоначальных исходных данных.

Используя следствие теоремы 2 получаем, что вычисление каждого выражения, не содержащего дополнительно определяемых функций, принадлежит классу **FP**. Аналогичное утверждение будет верно и относительно полиномиального числа присваиваний, включая и косвенную адресацию. ■

Литература

1. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.

2. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

3. *Косовская Т.М.* Машины Тьюринга // Компьютерные инструменты в образовании, 2005. № 3. С. 52–61.

4. *Косовская Т.М.* О временной сложности решения задач распознавания // Компьютерные инструменты в образовании, 2005. № 4. С. 61–68.

5. *Косовская Т.М.* О классах сложности алгоритмов // Компьютерные инструменты в образовании, 2005. № 5. С. 35–41.

6. *Косовский Н.К., Косовская Т.М.* Полиномиальность и NP-трудность задачи вычисления знака постоянного терма // Математические вопросы кибернетики, 2007. Вып. 16. С. 125–128.

7. *Форсайт Р.* Паскаль для всех. М.: Машиностроение, 1986.

8. *Du D.Z., Ko K.I.* Theory of Computational Complexity. A Wiley-Interscience Publication. John Wiley & Sons, Inc. 2000.

Abstract

The paper is devoted to the problem of proving that some function is in **FP** complexity class. Such proofs may use pascal-like functions, that are more easy for programmers, than computation models based on a Turing machine. It will be presented the idea of proof of the theorem, that if both the number of steps of pascal-like function evaluation and a size of all intermediate evaluations are not greater than some polynomial of the length of initial data, then this function is in **FP** and vice versa.

Keywords: P, **FP**, **NP**, Turing machine, pascal-like functions, NP-full, NP-complete, NP-complete functions examples.

Косовская Татьяна Матвеевна,
кандидат физико-математических наук, доцент кафедры математики Государственного Морского Технического Университета,
kosov@NK1022.spb.edu,

Косовский Николай Кириллович,
доктор физико-математических наук, профессор, заведующий кафедрой информатики математико-механического факультета Санкт-Петербургского государственного университета,
kosov@NK1022.spb.edu