Косовская Татьяна Матвеевна,
Косовский Николай Кириллович

ПРИНАДЛЕЖНОСТЬ КЛАССУ FP ДВАЖДЫ ПОЛИНОМИАЛЬНЫХ ПАСКАЛЕВИДНЫХ ФУНКЦИЙ НАД ПОДПРОГРАММАМИ ИЗ FP

Аннотация

В [3] даны определения числа шагов и длины записи промежуточных вычислений паскалевидных функций. С их помощью было получено представление класса всех функций, вычислимых на машинах Тьюринга за число шагов, ограниченное сверху полиномом от длины записи исходных данных (класса **FP**), с помощью дважды полиномиальных (то есть полиномиальных по числу шагов и по длине записи промежуточных вычислений) паскалевидных функций.

Для удобства доказательства принадлежности паскалевидной функции классу **FP** здесь введено понятие паскалевидных функций над списком *S* паскалевидный функций. Для этих функций даны определения числа шагов и длины записи промежуточных вычислений, наконец, определены дважды полиномиальные паскалевидные функции над списком *S*. (для каждой функции из *S* число шагов и длина записи промежуточных вычислений равны единице). Функции из *S* соответствуют базовым подпрограммам. Доказывается теорема о совпадении класса дважды полиномиальных паскалевидных функций над *S* и класса **FP**.

Ключевые слова: верхние оценки числа шагов, машины Тьюринга, класс **FP**, функции языка Паскаль.

ВВЕДЕНИЕ

Эта статья является продолжением статьи [3]. Однако она может читаться и независимо от нее. Она направлена на упрощение для программистов-математиков доказательств принадлежности классу **FP** алгоритмов, реализованных средствами языка Паскаль (паскалевидными функциями).

Полиномиальное число шагов для разных математических понятий алгоритмов не обязательно обеспечивает его принадлежность классу **FP** (классу функций, которые могут быть реализованы на машине Тьюринга, число шагов которой не превосходит полинома от длины записи исходных данных). В то же время программирование на машине Тьюринга [1, 2] мало адекватно современной практике программирования.

В интернете время от времени появляются «доказательства» того, что **P = NP** («решение» одной из семи задач, признанных в 2000 году сложнейшими задачами в математике). Как правило, ошибки в таких «доказательствах» являются следствиями того, что при подсчете числа шагов алгоритмов берется математическая модель алгоритма, для которой не доказа-

но, что из полиномиальности числа шагов его работы следует полиномиальность числа шагов машины Тьюринга, вычисляющего ту же функцию (или предикат).

В связи с этим в [3] было введено математическое понятие целочисленного алгоритма на основе широко распространенного языка Паскаль. Более точно, в основе рассматриваемого понятия лежит предложенный Н. Виртом язык Паскаль [4] с включением возможности описания и использования 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 над заданным списком базовых подпрограмм является определение того, что же считать шагом вычисления и что считать длиной используемой памяти. С определением шага вычисления, как правило, трудностей не возникает. Впрочем, на разных моделях вычисления функция 2^{2^n} вычисляется за очень разное число шагов, в зависимости от того, какие функции считаются исходными при ее вычислении.

В [3] даны определения числа шагов и длины записи промежуточных вычислений паскалевидных функций. С их помощью было получено представление класса функций, полиномиально быстро вычислимых на машинах Тьюринга (класса **FP**) с помощью дважды полиномиальных (то есть полиномиальных по числу шагов и по длине записи промежуточных вычислений) паскалевидных функций.

Для удобства доказательства принадлежности паскалевидной функции классу **FP** здесь введены определения числа шагов и длины записи промежуточных вычислений и, наконец, дважды полиномиальных паскалевидных функций над списком *S* паскалевидных функций (эти три определения не учитывают сложностные характеристики вычисления внутри всех функций из *S*). Функции из *S* соответствуют базовым подпрограммам.

В частности, для каждой функции из списка *S* число шагов над *S* (без учета их числа внутри вычисления функций списка *S*) считается ровно за один шаг.

Длина записи промежуточных вычислений над *S* (без учета длины записи внутри вычислений функций из списка *S*) отличается от длины записи вычислений паскалевидной функции тем, что не учитывается длина записи промежуточных внутренних вычислений всех функций из *S*. В частности, длина записи промежуточных вычислений над *S* (без учета их внутри вычислений функций из списка *S*) для каждой функции из *S* совпадает с максимумом длины записи значения функции и суммы длин всех ее аргументов.

Опишем это более подробно.

1. ОПРЕДЕЛЕНИЕ ШАГОВ ВЫЧИСЛЕНИЯ И ДЛИНЫ ЗАПИСИ ПРОМЕЖУТОЧНЫХ ВЫЧИСЛЕНИЙ НАД СПИСКОМ ПАСКАЛЕВИДНЫХ ФУНКЦИЙ

Определение. *Назовем термом над списком S паскалевидных функций любое арифметическое или булево выражение, возможно содержащее функции из S.*

Очевидно, что каждый терм однозначно задает некоторую функцию, в частности, терм над списком *S* паскалевидных функций задает паскалевидную функцию при использовании оператора присваивания имени функции, задаваемой этим термом. При этом процесс вычисления терма

t может быть представлен в виде корневого дерева. Глубиной записи *t* назовем глубину этого дерева.

При вычислении паскалевидной функции используются: оператор присваивания терма над списком паскалевидных функций, условные операторы, операторы цикла и рекурсивные вызовы процедур и функций.

Дадим два совместно рекурсивно задаваемых определения.

Определение. *Длиной записи вычисления терма t на глубине i над списком S паскалевидных функций назовем сумму длин абсолютных величин значений всех подвыражений глубины i выражения t, в том числе и содержащего функции из S. Для вновь определяемых процедур и функций, входящих в терм и не входящих в S, берется длина записи промежуточных вычислений таких функций, включая все их рекурсивные вызовы.*

Длиной записи вычисления выражения t над S назовем максимум по всем i (от 1 до глубины t) длин записи вычисления t на глубине i над S.

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

Под длиной записи промежуточных вычислений (то есть используемой памяти) над списком S паскалевидных функций понимаем максимум длины записей вычисления по всем шагам этого вычисления над S, включая начальный и заключительный, но не учитывая внутренних шагов вычисления функции из S.

Для удобства читателя предполагаем, что все числа записаны в десятичной системе счисления (хоть основание системы счисления может быть любым положительным целым числом, отличным от единицы).

Таким образом, в приведенных определениях не учитываются рекурсивные вызовы подпрограмм из *S*, используемые для вычисления паскалевидной функции, и длина записи внутренних промежуточных вычислений в каждой такой подпрограмме из *S*.

Фактически здесь определены число шагов паскалевидной функции и длина записи ее промежуточных вычислений без учета внутренних шагов и внутренних промежуточных вычислений функций из конечного списка *S*, являющихся паскалевидными функциями.

Итак, в определенное здесь число шагов над *S* (без учета числа шагов внутри вычисления функций из *S*) вызов каждой функции из *S* добавляет только единицу. К длине записи промежуточных вычислений над *S* (также без учета их внутри вычисления функций из *S* вызов каждой функции из *S* добавляет максимум длины записи результата и суммы длин записи всех аргументов при каждом обращении к функции из *S*.

Две введенные здесь характеристики сложности вычисления паскалевидной функции над *S* измеряются относительно длины записи исходных данных.

2. ОСНОВНОЙ РЕЗУЛЬТАТ

Определение. *Дважды полиномиальной паскалевидной функцией над конечным списком паскалевидных функций S назовем такую паскалевидную функцию, для которой как ее число шагов, так и ее длина записи промежуточных данных (без учета их внутри вычисления функций из S) не превосходят некоторого полинома от длины записи исходных данных.*

Теорема. *Класс паскалевидных функций (в том числе и класс функций, задающих предикаты), дважды полиномиальных над списком S, все функции из которого принадлежат классу* **FP**, *совпадает с классом* **FP** *(соответственно с классом* **P***).*

Схема доказательства. Ленту машины Тьюринга легко промоделировать двумя линейными динамическими массивами, верхняя граница которых может быть задана ограничением на длину используемой ленты.

Докажем, что всякая дважды полиномиальная паскалевидная функция, удовлетворяющая условию теоремы, 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. В этом случае косвенная адресация становится полиномиально ограниченной сверху по числу шагов машины Тьюринга.

Прежде всего 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 из [3] получаем, что вычисление каждого выражения, не содержащего дополнительно определяемых функций, принадлежит классу **FP**. Аналогичное утверждение будет верно и относительно полиномиального числа присваиваний, включающих и косвенную адресацию.

Поскольку имеется полином от длины записи исходных данных, ограничивающий сверху значение всех вычисляемых выражений и подвыражений, встречающихся во время вычисления, то число шагов их вычисления ограничено сверху одним и тем же (но другим, как правило, большей степени) полиномом. Сумма полиномиального числа слагаемых, являющихся полиномами, также является полиномом. Поэтому суммируя все шаги, используемые для вычисления всех выражений, получаем общую полиномиальную оценку числа шагов их вычисления на машине Тьюринга. Все встроенные операции, включая косвенную адресацию, осуществляются за полиномиально ограниченное сверху число шагов.

Рекурсивные процедуры и паскалевидные функции могут быть реализованы на PAM [1] традиционным для программирования способом с помощью стека. В этом стеке при каждом рекурсивном вызове записываются значения всех внутренних переменных, получивших к этому моменту какие-либо значения.

При этом память ограничена сверху (в том числе и в совокупности присвоенных значений элементам динамических массивов) полиномом от числа выполненных операторов присваиваний, умноженному на полином от длины записи исходных данных, являющемуся верхней границей длин записи всех значений переменных и элементов массива.

Поэтому справедливо следующее утверждение. Каждое присваивание (в том числе и элементу массива), включая присваивание аргументов при рекурсивных вызовах функций, принадлежит **FP**.

Заметим, что в случае присваивания элементу массива вычисляется пара из совокупности всех индексов и значения этого элемента массива.

Поэтому результаты вычисления любой последовательности полиномиального числа присваиваний с такими же ограничениями на длину записи промежуточных вычислений могут быть реализованы на машине Тьюринга за число шагов, ограниченное сверху некоторым полиномом. ▪

Доказанная теорема, по существу, устанавливает замкнутость класса **FP** относительно задания паскалевидной функции, являющейся дважды полиномиальной над списком подпрограмм (паскалевидных функций) из **FP**.

Определение. *Назовем возвратом при вычислении паскалевидной функции над S выполнение любого из следующих операторов: goto назад по тексту программы, повторное выполнение тела цикла, вызов функции из S и, наконец, вызов (в том числе и рекурсивный) функции не из S или вызов процедуры (в том числе и рекурсивный).*

В определении дважды полиномиальной паскалевидной функции слова «число шагов» можно заменить словами «число возвратов». При этом утверждение теоремы останется верным. Уточним вариант формулировки и докажем его в качестве следствия.

Следствие. *Если при вычислении паскалевидной функции f над списком S, все функции которого принадлежат* **FP**, *число возвратов и длина использованной памяти над S не превосходят полинома от длины записи ее исходных данных, то эта функция принадлежит* **FP**.

Доказательство. Число шагов функции f над S не превосходит числа возвратов над S, умноженного на число всех операторов, включая и все подоператоры определения функции f. Осталось применить теорему. ∎

Доказанное следствие существенно упрощает доказательство принадлежности паскалевидной функции как классу **FP**, так и классу **P**.

Теорема и ее следствие позволяют сравнивать между собой паскалевидные функции из **FP** по трем существенным характеристикам: числу шагов вычисления над списком S, числу возвратов над списком S и длине записи промежуточных вычислений над списком S, в том числе и над пустым списком S.

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., Косовский Н.К.* Основы доказательств полиномиальной быстроты простейших математических алгоритмов // Компьютерные инструменты в образовании, 2010. № 2. С. 3–13.

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

Abstract

[3] gives definitions of time and space comlexity of pascal-like functions. Using this definitions it was obtained the representation of the class of all functions computable by Turing machine in time upper bounded by the polynomial of the length of the initial data (**FP** class) by means of double polynomial (i.e. polynomial in time and space) pascal-like functions.

For the convinience of proving that a pascal-like function is in **FP**, we introduce the notion of the pascal-like functions over a list S of pascal-like functions. For such functions we give definitions of time and space complexities, finally, we define double polynomial pascal-like functions over a list S. (for every function in S, time and space complexity are equal to one). Functions from S correspond to basic subroutines. The theorem about the equality of the class of double polynomial pascal-like functions over S and the **FP** class is proved.

Keywords: Upper bounds for time complexity, Turing machine, **FP** complexity class, functions of pascal programming language.

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