Введение в рекурсивные функции

Исторически первой алгоритмической системой была система, основанная на использовании конструктивно определяемых арифметических (целочисленных) функций, получивших специальное название рекурсивных функций. В математических исследованиях, посвященных алгоритмам, данная система имеет наибольшее распространение.

Рекурсией называется способ задания функции, при котором значение определяемой функции для произвольных значений аргументов выражается известным образом через значения определяемой функции для меньших значений аргументов.

В процессе построения теории численных алгоритмов выяснилось, что любой алгоритм можно простыми методами свести к численному. Таким образом, теория численных алгоритмов («теория вычислимых функций») стала универсальным аппаратом для исследования алгоритмических проблем.

Численные функции, значение которых можно установить посредством некоторого алгоритма, называются **вычислимыми функциями**. Функция называется **рекурсивной**, если существует эффективная процедура для ее вычисления. Понятие эффективной процедуры является интуитивным. Говорят, что имеется эффективная процедура для выполнения определенных вычислений, если эти вычисления выполняются по механическим правилам, т. е. по определенному алгоритму.

Поскольку понятие алгоритма в этом определении берется в интуитивном смысле, то и понятие вычислимой функции оказывается интуитивным. Тем не менее, при переходе от алгоритмов к вычислимым функциям возникает одно очень существенное обстоятельство: совокупность процессов, подпадающих под интуитивное понятие алгоритма, очень обширна и мало обозрима. Совокупность вычислимых функций для самых разных пониманий процессов оказалась одной и той же и притом легко описываемой в обычных математических терминах.

Совокупность числовых функций, совпадающая с совокупностью всех вычислимых функций при самом широком до сих пор известном понимании алгоритма, носит название совокупности рекурсивных функций.

К. Гёдель впервые описал класс всех рекурсивных функций как класс всех числовых функций, определимых в некоторой формальной системе. Исходя из совершенно других предпосылок, Черч в 1936 г. вывел тот же класс числовых функций, что и Гёдель.

Историческая справка.

Курт Гёдель

Курт Фри́дрих Гёдель (нем. *Kurt Friedrich Gödel*; 28 апреля 1906, Брюнн, Австро-Венгрия — 14 января 1978, Принстон, Нью-Джерси) — австрийский логик, математик и философ математики.

Наиболее известен сформулированными и доказанными им теоремами о неполноте, которые оказали огромное влияние на представление об основаниях математики. Считается одним из наиболее выдающихся мыслителей XX века

Черчем была сформулирована гипотеза о том, что *класс рекурсивных функций тождествен с классом всюду определенных вычислимых функций*. Эта гипотеза известна под именем **тезиса Черча.**

Понятие вычислимой функции точно не определяется, поэтому тезис Черча доказать нельзя.

Историческая справка.

Алонзо Черч

Алонзо Чёрч (англ. Alonzo Church; 14 июня 1903 года, Вашингтон — 11 августа 1995 года, Хадсон, Огайо, США) — американский математик и логик, внесший значительный вклад в основы информатики.

Получил степень бакалавра искусств в Принстонском университете в 1924 году, и докторскую (Ph.D.) в 1927 году под руководством Освальда Веблена за работу «Alternatives to Zermelo's Assumption». Два года он был нацисследовательским стипендиатом (National Research Fellow), год провёл в Гарварде, затем — в Геттингене и Амстердаме. С 1929 года ассистент-профессор математики в альма-матер, с 1939 года доцент, с 1947 года профессор математики, с 1961 года профессор математики и философии.

Чёрч прославился разработкой теории лямбда-исчислений, последовавшей за его знаменитой статьёй 1936 года, в которой он показал существование т. н. «неразрешимых задач» (теорема Чёрча-Тьюринга). Эта статья предшествовала знаменитому исследованию Алаа Тьюринга на тему проблемы остановки, в котором также было продемонстрировано существование задач, неразрешимых механическими способами. Впоследствии Чёрч и Тьюринг показали, что лямбда-исчисления и машина Тьюринга имели одинаковые свойства, таким образом, доказывая, что различные «механические процессы вычислений» могли иметь одинаковые возможности. Эта работа была оформлена как тезис Чёрча-Тьюринга.

Помимо прочего, его система лямбда-исчислений легла в основу функциональных языков программирования, в частности семейства Лисп (например, Scheme).

Если некоторым элементам множества X поставлены в соответствие однозначно определенные элементы множества Y, то говорят, что задана **частичная функция** из X в Y.

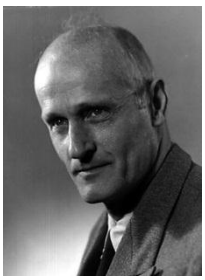
Совокупность тех элементов множества X, у которых есть соответствующие в Y, называется **областью определенности** функции, а совокупность этих элементов множества Y – **совокупностью значений** функции.

Если область определенности функции из X в Y совпадает с множеством X, то функция называется **всюду определенной.**

Клини ввел понятие частично рекурсивной функции и высказал гипотезу, что все частичные функции, вычислимые посредством алгоритмов, являются частично рекурсивными. Эта гипотеза также недоказуема, как и гипотеза Черча. Однако математические исследования последних 30 лет выявили полную целесообразность считать понятие частично рекурсивной функции научным эквивалентом интуитивного понятия вычислимой частичной функции.

В дальнейшем под тезисом Черча будем понимать гипотезу Черча в том расширенном виде, который был придан ей Клини.

Историческая справка.



Стивен Коул Клини

Сти́вен Ко́ул Кли́ни (правильнее — Кле́йни, англ. Stephen Cole Kleene; 5 января 1909, Хартфорд, Коннектикут, США — 25 января 1994, Мадисон, Висконсин, США) — американский математик и логик.

Его работы совместно с работами Алонзо Чёрча, Курта Гёделя и Алана Тьюринга дали начало разделу математической логики — теории вычислимости. Кроме того, известен изобретением регулярных выражений. Его именем названы Алгебра Клини, Звёздочка Клини, теорема Клини о рекурсии, теорема Клини о неподвижной точке. Работал также в области интуиционистсткой математики Брауэра. Внёс важный вклад в теорию конечных автоматов (см. теорема Клини).

Сам Клини произносил свою фамилию как «Клейни», ошибочная транслитерация «Клини» утвердилась в Советском Союзе в связи с изданием переводов его книг именно под такой фамилией.

Среди наиболее известных работ, издававшихся на русском языке — книги «Введение в метаматематику» и «Математическая логика».

Тезис Черча оказался достаточным, чтобы придать необходимую точность формулировке алгоритмических проблем и в ряде случаев сделать возможным доказательство их неразрешимости. В силу тезиса Черча вопрос о вычислимости функции равносилен вопросу о ее рекурсивности. Понятие рекурсивной функции строгое. Поэтому обычная математическая техника позволяет иногда непосредственно доказать, что решающая задачу функция не может быть рекурсивной, т. е. задача неразрешима.

Применение рекурсивных функций в теории алгоритмов основано на идее нумерации слов в произвольном алфавите последовательными натуральными числами. Наиболее просто такую нумерацию можно осуществить, располагая слова в порядке возрастания их длин, а слова, имеющие одинаковую длину, — в произвольном (лексикографическом) порядке.

ТЕОРЕМА. Любой алгоритм можно свести к вычислению значений некоторой целочисленной функции при целочисленных значениях аргументов.

Доказательство. Включим все условия задачи, доступные для переработки данным алгоритмом A в пронумерованную неотрицательными целыми числами последовательность A_0, A_1, A_2, …, A_n, …

Аналогично, записи возможных решений также включим в пронумерованную последовательность B_0, B_1, B_2, …, B_m, …

После проведения нумерации очевидно, что любой алгоритм, перерабатывающий запись условий A_n в запись решений B_m, можно свести к вычислению значений некоторой числовой функции $m=\varphi(n)$, так как после проведения нумерации можно иметь дело лишь с соответствующими номерами записей условий и решений, а не с самими записями. ***Теорема доказана.***

Теперь можно говорить об алгоритме, перерабатывающем номер записи условия в номер записи решения, который будет численным алгоритмом.

Очевидно, что при наличии алгоритма, решающего исходную задачу, имеется и алгоритм, вычисления значений соответствующей функции.

Действительно, чтобы найти значение $\varphi(n)$ при $n=n^*$ (* означает конкретное значение n), можно по n^* восстановить запись условия задачи, а затем по имеющемуся алгоритму найти запись решения и по ней определить номер $m=m^*$, т.е. $\varphi(n^*)=m^*$.

Наоборот, если есть алгоритм вычисления функции $\varphi(n)$, то имеется и алгоритм решения исходной задачи, т.к. по записи условия задачи можно найти соответствующий ей номер n^*, затем вычислить $m^*=\varphi(n^*)$ и по m^* определить запись решения.

Один из методов такой нумерации был предложен австрийским математиком Куртом Гёделем (род. 1906 г.). Любое целое неотрицательное число n можно представить в форме $n= P_0^{a_1} \cdot P_1^{a_2} \cdot P_2^{a_3} \cdot \ldots \cdot P_{m-1}^{a_m}$, где $P_0=2$, $P_1= 3, 5, \ldots, P_{m-1}$, т.е. P_i – i-е простое число.

В силу теоремы до единственности разложения любого числа на простые множители следует, что каждому числу n однозначно соответствует набор $A=\{a_1, a_2, \ldots a_m\}$, и, наоборот, каждому набору A однозначно соответствует число n.

Например, если $n = 60$, имеем $60=2^2 \cdot 3^1 \cdot 5^1$, т.е. $a_1=2$, $a_2=1$, $a_3=1$.

Если n = 98, тогда $98=2^1\cdot3^0\cdot5^0\cdot7^2$, т.е. $a_1=1$, $a_2=0$, $a_3=0$, $a_4=2$.

С помощью этого способа (**гёделизации**) можно нумеровать любые упорядоченные последовательности, состоящие из m чисел.

Приведем несколько примеров.

Пример 1. Каждой паре чисел a_1 и a_2, для которых мы ищем НОД = q, можно поставить в соответствие гёделевский номер этой пары $n=2^{a_1}\cdot3^{a_2}$. Тогда алгоритм Евклида сведется к вычислению функции $q=\varphi(n)$.

Пример 2. Пусть требуется перенумеровать все слова в некотором алфавите А. Это легко сделать, поставив каждой букве алфавита в соответствие какое-либо число. тогда каждому слову в алфавите А Будет соответствовать последовательность чисел (номеров букв). Осталось только вычислить гёделевский номер этой последовательности. Возьмем для определенности алфавит русских букв А={а,б,в,г,д,е,ж,з,и,…э,ю,я} и пронумеруем их числами 1, 2, 3, 4, 5, 6, 7, 8, … Вычислим гёделевские номера некоторых русских слов. Слову “бег” соответствует набор {2, 6, 4}, т.е. $n = 2^2\cdot3^6\cdot5^4 = 4\cdot729\cdot625 = 1822500$; слову “баба” соответствует набор {2, 1, 2, 1}, т.е $n = 2^2\cdot3^1\cdot5^2\cdot7^1 = 4\cdot3\cdot25\cdot7 = 2100$.

Самостоятельно рассмотреть вопрос о возможности нумерации букв алфавита последовательностью 0, 1, 2, …

Заметим, что не только арифметические алгоритмы сводятся к вычислению значений целочисленных функций. Любой нормальный алгоритм Маркова может быть также сведен к вычислению значений целочисленных функций. Поэтому алгоритмы вычисления целочисленных функций можно считать универсальной формой алгоритма.

РЕКУРСИВНЫЕ ФУНКЦИИ

Формализация понятия алгоритма (вычислимой функции) в терминах рекурсивной функции была предложена в середине 1930-х годов в работах Черча («Общерекурсивные функции», 1936 год) и Клини («Частично-рекурсивные функции», 1938 год), которые основывались на более ранних работах Геделя и Эрбрана.

Общие сведения

Рекурсией называется способ задания функции, при котором значение определяемой функции для произвольных значений аргументов выражается известным образом через значения определяемой функции для меньших значений аргументов.

Применение рекурсивных функций в теории алгоритмов основано на идее нумерации слов в произвольном алфавите последовательными натуральными числами. Наиболее просто такую нумерацию можно осуществить, располагая слова в порядке возрастания их длин, а слова, имеющие одинаковую длину, - в произвольном порядке.

После нумерации входных и выходных слов в произвольном алфавитном операторе этот оператор превращается в функцию $y=f(x)$, в которой аргумент x и функция y принимают неотрицательные целочисленные значения. Функция $f(x)$ может быть определена не для всех значений x, а лишь для тех, которые составляют область определения этой функции.

Понятие простейших функций

Числовые функции, значение которых можно установить посредством некоторого алгоритма, называются *вычислимыми функциями.*

Для того чтобы описать класс функций с помощью рекурсивных определений, рассмотрим набор простейших (базовых) функций:

1) $Z(x_1, x_2, \ldots, x_n) = 0$ — нуль-функция, которая определена для всех неотрицательных значений аргумента;
2) $s(x) = x+1$ — функция непосредственного следования, также определенная для всех целых неотрицательных значений своего аргумента;
3) $I_m^n = (x_1, \ldots, x_m, \ldots, x_n) = x_m$ — функция выбора (тождества), повторяющая значения своих аргументов.

Используя простейшие функции в качестве исходных функций, можно с помощью небольшого числа общих конструктивных приемов строить сложные арифметические функции. В теории рекурсивных функций особо важное значение имеют три операции: **суперпозиции, примитивной рекурсии и минимизации.**

Оператор суперпозиции

Оператором суперпозиции S называется подстановка в функцию от m переменных m функций от n одних и тех же переменных. Она дает новую функцию от n переменных.

Например, из функций $f(x_1, x_2, \ldots, x_m)$, $f1(x_1, x_2, \ldots, x_n)$, $f2(x_1, x_2, \ldots, x_n)$, …, $fm(x_1, x_2, \ldots, x_n)$ можно получить новую функцию:

$Sm+1(f, f_1, f_2, \ldots, f_m) = g(x_1, x_2, \ldots, x_n) = f(f1(x_1, x_2, \ldots, x_n), f2(x_1, x_2, \ldots, x_n), \ldots, fm(x_1, x_2, \ldots, x_n))$

При помощи оператора суперпозиции и функции выбора можно выразить любую подстановку функции в функцию.

Например, осуществляя операцию суперпозиции функций $f(x) = 0$ и $g(x) = x+1$, получим функцию $h(x) = g(f(x)) = 0 + 1 = 1$

При суперпозиции функции g(x) с этой же функцией получим функцию $h(x) = g(g(x)) = x+2$.

Оператор примитивной рекурсии

Оператор примитивной рекурсии *Rn* позволяет определить (n+1)-местную функцию f по двум заданным функциям, одна из которых является *n*-местной функцией g, а другая (*n+2*)-местной функцией *h.*

Функция $f(x_1, x_2, \ldots, x_n, y)$ получается оператором примитивной рекурсии из функций $g(x_1, x_2, \ldots, x_n)$ и функции $h(x_1, x_2, \ldots, x_n, y, z)$, если:

$$\mathbf{f(x_1, x_2, \ldots, x_n, 0) = g(x_1, x_2, \ldots, x_n);}$$

$$\mathbf{f(x_1, x_2, \ldots, x_n, y+1) = h(x_1, x_2, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y)). (*)}$$

Приведенная пара равенств (*) называется схемой примитивной рекурсии. Для понимания операции примитивной рекурсии необходимо заметить, что всякую функцию от меньшего числа аргументов можно рассматривать как функцию от большего числа аргументов. Существенным в операторе примитивной рекурсии является то, что независимо от числа переменных в f рекурсия ведется только по одной переменной y. Остальные n переменных x_1, x_2, …, x_n на момент применения схемы (*) зафиксированы и играют роль параметров.

Оператор минимизации

Отношение $P(x_1, x_2, \ldots, x_n)$ называется примитивно-рекурсивным, если примитивно-рекурсивна его характеристическая функция:

$$X(x_1, x_2, \ldots, x_n) = \begin{cases} 0, \text{если } P(x_1, x_2, \ldots, x_n) - \text{"ложь"} \\ 1, \text{если } P(x_1, x_2, \ldots, x_n) - \text{"истина"} \end{cases}$$

Предикат называется *примитивно-рекурсивным*, если его характеристическая функция примитивно-рекурсивна.

Функция $f(x_1, x_2, \ldots, x_n)$ получается посредством операции минимизации из предиката $P(x_1, x_2, \ldots, x_n, z)$, если в любой точке значением функции $f(x_1, x_2, \ldots, x_n)$ является минимальное значение z, обращающее предикат $P(x_1, x_2, \ldots, x_n, z)$ в значение «истина»:

$f(x_1, x_2, \ldots, x_n) = \mu z(P(x_1, x_2, \ldots, x_n, z))$, где μz – оператор минимизации.

Ограниченный оператор минимизации

Функция $f(x_1, x_2, \ldots, x_n)$ получается ограниченным оператором минимизации из предиката $P(x_1, x_2, \ldots, x_n, z)$ и функции $U(x_1, x_2, \ldots, x_n)$, если в любой точке значение этой функции определено следующим образом:

1) для любого $z < U(x_1, x_2, \ldots, x_n)$, такого, что $P(x_1, x_2, \ldots, x_n, z)$ = «истина», значение функции $f(x_1, x_2, \ldots, x_n) = \mu z(P(x_1, x_2, \ldots, x_n, z))$;
2) не для любого $z < U(x_1, x_2, \ldots, x_n)$, такого, что $P(x_1, x_2, \ldots, x_n, z)$ = «истина», значение функции $f(x_1, x_2, \ldots, x_n) = U(x_1, x_2, \ldots, x_n)$.

Данное определение записывается следующим образом:

$f(x_1, x_2, \ldots, x_n) = \mu z < U(x) (P(x_1, x_2, \ldots, x_n, z))$.

Ограничение z в ограниченном операторе минимизации дает гарантию окончания вычислений, поскольку оно оценивает сверху число вычислений предиката P. Возможность оценить сверху количество вычислений является существенной особенностью примитивно-рекурсивных функций.

Примитивно-рекурсивные и частично-рекурсивные функции

Большинство вычислимых функций относится к классу примитивно-рекурсивных функций.

Функция называется *примитивно-рекурсивной*, если она может быть получена из простейших функций с помощью конечного числа операторов суперпозиции и примитивной рекурсии.

Если некоторые функции являются примитивно-рекурсивными, то в результате применения к ним операторов суперпозиции или примитивной рекурсии можно получить новые примитивно-рекурсивные функции.

Существует три возможности доказательства того, что функция является примитивно-рекурсивной:

а) показать, что заданная функция является простейшей;
б) показать, что заданная функция построена с помощью оператора суперпозиции;
в) показать, что заданная функция построена с помощью оператора примитивной рекурсии.

Тем не менее примитивно-рекурсивные функции не охватывают все вычислимые функции, которые могут быть определены конструктивно. При построении этих функций могут использоваться другие операции. Функция называется *частично-рекурсивной,* если она может быть получена из простейших функций с помощью конечного числа операторов суперпозиции, примитивной рекурсии и минимизации. Указанные операции могут быть выполнены в любой последовательности и любое конечное число раз.

Если такие функции являются всюду определенными, то они называются общерекурсивными функциями. Частично-рекурсивные функции вычислимы путем определенной процедуры механического характера. Практически понятием частично-рекурсивных функций пользуются для доказательства алгоритмической разрешимости или неразрешимости проблем.

Приведем тезис Черча, который связывает понятие алгоритма и строгое математическое понятие частично-рекурсивной функции.

Тезис Черча: *всякий алгоритм может быть реализован частично-рекурсивной функцией.*

Утверждение о не существовании частично-рекурсивной функции эквивалентно не существованию алгоритма решения соответствующей задачи.

ТИПЫ РЕКУРСИВНЫХ АЛГОРИТМОВ

Эффективность разработки рекурсивного алгоритма определяется наличием некоторых условий:

1) если исходные данные имеют рекурсивную структуру, то процедуры анализа таких структур будут более эффективны, если они сами рекурсивны;

2) если алгоритм обработки некоторого набора данных построить, разбивая данные на части и обрабатывая в отдельности каждую часть, то из частичных решений можно получить общее;

3) если по условию задачи необходимо выбрать оптимальный вариант из некоторого множества решений, то искомое решение может быть найдено через конечное число шагов.

Поиск решения завершается после окончания данных либо при нахождении искомого решения на текущем наборе данных.

Примеры решения задач

ПРИМЕР 1. Доказать примитивную рекурсивность функции f(x, y) = x + y.

Решение. Рассмотрим способ рекурсивного определения данной функции:

f(x, 0) = x,

f(x, y+1) = x + y + 1 = f(x, y) +1

Чтобы доказать соответствие данной рекурсивной схемы схеме примитивной рекурсии воспользуемся функциями выбора и следования:

f(x, 0) = x = I(x),

f(x, y+1) = f(x, y) +1 = S(f(x, y)) = S ($I_3^3(x, y, f(x, y))$)

ПРИМЕР 2. Доказать примитивную рекурсивность функции f(x, y) = x · y.

Решение. Рассмотрим способ рекурсивного определения данной функции:

f(x, 0) = x,

f(x, y+1) = x · (y + 1) = x · y + x = f(x, y) +x, из которого следует, что Z(x) = x·0

Обозначим x · y = p(x,y), тогда:

p(x, 0) = Z(x);

p(x, y+1) = p(x, y) + x = S(p(x, y), x) = S ($I_1^3\big(x, y, p(x, y)\big), I_3^3(x, y, p(x, y))$)

Вопросы для самоконтроля

1. Какие функции считаются базисными?
2. В чем особенность действия оператора суперпозиции?
3. В чем особенность действия оператора примитивной рекурсии?
4. Что такое примитивно-рекурсивная функция?
5. Что такое примитивно-рекурсивный оператор?
6. Как действует оператор минимизации?
7. Сформулируйте определение частично-рекурсивной функции.
8. Может ли примитивно-рекурсивная функция быть частично-рекурсивной?
9. Проверьте, выполняются ли основные требования к алгоритмам в классе частично-рекурсивных функций.
10. Когда частично-рекурсивная функция называется общерекурсивной?
11. Всякая ли функция, вычислимая некоторым алгоритмом, частично-рекурсивна?

Упражнения

Доказать примитивную рекурсивность следующих функций.

1. Нахождение минимального значения из двух заданных чисел f(x, y)=min(x, y).
2. Нахождение максимального значения из двух заданных чисел f(x, y)=max(x, y).
3. Функция r(x, y) – остаток от деления x на y.
4. Функция q(x, y) – частное от деления x на y.
5. Доказать, что предикат «x делится на n» примитивно рекурсивен для любого n.
6. Доказать, что предикат «x делится на n и m» примитивно рекурсивен для любых n и m.

7. Доказать, что отношение $x_1 > x_2$ примитивно-рекурсивно.
8. Доказать, что предикат f(x) = g(x) примитивно-рекурсивен, если функции f(x) и g(x) примитивно-рекурсивны.
9. Дано множество слов одинаковой длины, причем первые два слова выделены. Построить цепь от первого слова ко второму так, чтобы все слова этой цепи были только из заданного множества. Соседние слова построенной цепи должны отличаться только одной буквой.
10. Дано множество слов. Построить из них кроссворд заданной конфигурации (число слов может быть больше требуемого количества для заполнения кроссворда).

ЗАКЛЮЧЕНИЕ

Всякий алгоритм можно рассматривать как некоторое универсальное средство для решения целого класса задач. Но существуют такие классы задач, для решения которых нет общего универсального алгоритма. Проблемы решения такого рода называются алгоритмически неразрешимыми. Однако алгоритмическая неразрешимость проблемы решения задач того или иного класса не означает невозможность решения любой частной задачи из этого класса. Переход от интуитивного понятия алгоритма к формальному определению алгоритма позволяет доказать алгоритмическую неразрешимость ряда проблем.

СПИСОК ЛИТЕРАТУРЫ

1. Бильгаева, Н. Ц. Теория алгоритмов, формальных языков, грамматик и автоматов: учебное пособие [Текст]/ Н. Ц. Бильгаева. – Улан-Удэ : Изд-во ВСГТУ, 2000. – 51 с.
2. Верещагин, Н. К. Математическая логика и теория алгоритмов. Вычислимые функции [Текст]/ Н. К. Верещагин, А. Шень. – М.: МЦНМО, 2008. – 193 с.
3. Верещагин, Н. К. Начала теории множеств. Математическая логика и теория алгоритмов. [Текст]/ Н. К. Верещагин, А. Шень. – М.: МЦНМО, 2008. – 128 с.
4. Игошин, В. И. Задачи и упражнения по математической логике и теории алгоритмов [Текст]/ В. И. Игошин. – М.: Академия, 2007. – 304 с.
5. Матрос, Д. Ш. Теория алгоритмов: учебник [Текст]/ Д. Ш. Матрос, Г. Б. Поднебесова. – М.: БИНОМ. Лаборатория знаний, 2008. – 202 с.
6. Пономарев, В. Ф. Основы теории алгоритмов: учебное пособие [Текст]/ В. Ф. Пономарев. – Калининград: Изд-во КГТУ, 2005. – 53 с.
7. Фалевич, Б. Я. Теория алгоритмов: учебное пособие [Текст]/ Б. Я. Фалевич. – М.: Машиностроение, 2004. – 160 с.
8. Шелупанов, А.А. Математическая логика и теория алгоритмов : учебное пособие для вузов [Текст]/ А. А. Шелупанов, В. М. Зюзьков. – М.: Академия, 2008. – 176 с.