КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

В.Ф. ПОНОМАРЁВ

ОСНОВЫ ТЕОРИИ АЛГОРИТМОВ

Утверждено Ученым советом университета в качестве учебного пособия по дисциплине «Математическая логика и теория алгоритмов» для студентов специальностей
230101.65- Вычислительные машины, комплексы, системы и сети,
230102.65-Автоматизированные системы обработки информации и управления

Калининград
Издательство КГТУ
2005

КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

В.Ф. ПОНОМАРЁВ

ОСНОВЫ ТЕОРИИ АЛГОРИТМОВ

Утверждено Ученым советом университета в качестве учебного пособия по дисциплине «Математическая логика и теория алгоритмов» для студентов специальностей
230101.65- Вычислительные машины, комплексы, системы и сети,
230102.65-Автоматизированные системы обработки информации и управления

Калининград
Издательство КГТУ
2005

ББК 32.973
УДК 681.142

Пономарев В.Ф. Основы теории алгоритмов: учебное пособие. - Калининград: Изд-во КГТУ, 2005.- 53с.

Учебное пособие предназначено студентам, изучающим «Математическую логику и теорию алгоритмов». В нем рассмотрены три алгоритмических модели: рекурсивные функции, машина Тьюринга и нормальные алгоритмы Маркова. Эти модели составляют фундамент теории алгоритмов и служат основой для разработки и анализа вычислительных алгоритмов практических задач. На многочисленных примерах показаны процедуры вычисления простейших числовых функций. В заключении приведены оценки временно́й и ёмкостной сложности вычислительных алгоритмов.

Рис.-26, табл.-16, список лит.-10 название

Рецензент - Жебелев С.И., к.ф.-м. н., доцент кафедры систем управления и вычислительной техники Калининградского государственного технического университета

Вениамин Федорович Пономарев
ОСНОВЫ ТЕОРИИ АЛГОРИТМОВ

Предисловие

Понятие "алгоритм" принадлежит к числу основных понятий математики и занимает одно из центральных мест в организации вычислительных процессов и проектировании разнообразных дискретных устройств.

Еще в IX веке великий узбекский ученый Мухаммед бен-муса аль-Хорезми разработал систему правил четырех арифметических действий над числами в десятичной позиционной системе счисления (он жил приблизительно с 783 по 850г.). Эти правила предписывали строгую последовательность действий для получения искомого результата. Так возникло понятие "алгоритм" в честь арабского имени аль-Хорезми.

Позднее это название было принято для обозначения совокупности правил вычислительных процессов, в которых искомые результаты практических задач находят последовательно из исходных данных по определенным правилам и инструкциям.

Процесс создания компьютерной программы для решения какой-либо практической задачи включает в себя формализацию этой задачи, разработку вычислительного алгоритма её решения, написание и отладку программы, наконец, решение поставленной задачи.

Основными объектами алгоритма стали ***данные***. Для уточнения этого понятия фиксируют конечный алфавит символов (цифр, букв и т.п.) и правил построения ***алгоритмических объектов***. Такими объектами вычислительных алгоритмов являются списки и стеки, множества и отображения. Процесс преобразования алгоритмических объектов от исходных данных до искомого результата выполняется ***дискретно*** или, как говорят, "по шагам". Преобразование за один шаг носит локальный характер, т.е. этому преобразованию подвергается не весь объект, а лишь его часть: элемент стека или компонента кортежа, столбец или строка матрицы и т.п. Последовательность шагов детерминирована, т.е. после каждого шага указывается точно, что и как следует выполнять на следующем шаге. Процесс преобразования алгоритмического объекта, включающий в себя заданную последовательность шагов, называют ***алгоритмическим процессом***. Механизм реализации алгоритмического процесса удобнее проследить на ***алгоритмических моделях***, использующих конечные наборы простейших объектов и конечные наборы элементарных действий.

Выделяют три основных типа алгоритмических моделей:

первый — ***рекурсивные функции*** — связывает понятие алгоритма с элементарными вычислительными операциями на множестве целых положительных чисел;

второй — ***машина Тьюринга*** — связывает понятие алгоритма с механическим устройством, способным выполнять строго фиксированное множество элементарных действий над простейшими символами;

третий — ***нормальный алгоритм Маркова*** — связывает понятие алгоритма с элементарными преобразованиями слов произвольного алфавита, замещая части или всего слова другим словом.

В теории вычислительных алгоритмов доказана сводимость одного типа модели к другой, т.е. всякий алгоритм, описанный средствами одной модели, может быть описан также средствами другой.

Рассмотрим подробнее реализацию алгоритмических процессов для вычисления числовых функций на трех типах моделей алгоритмов.

Для закрепления полученных знаний рекомендуется самостоятельно решить задачи, представленных в «Вопросы и задачи» и сверить их с решениями, представленными в «Ответы и решения». Кроме того, рекомендуется выполнить индивидуальное задание по разработке программы вычисления простейших функций на машине Тьюринга.

1. Рекурсивная функция - модель алгоритма

Рекурсия[*] есть способ вычисления значения числовой функции по известным значениям независимых переменных аргумента и известному значению функции в некоторой исходной точке.

Известно, что любое число может быть синтаксически представлено цепочкой (<целое>"."<целое>), где существует только одна синтаксическая переменная — <целое>. Поэтому любую вычислимую функцию, заданную на множестве натуральных чисел и принимающую значения на том же множестве принято называть ***рекурсивной функцией***[*].

Если значения функции найдены не для всех значений области определения, то её называют ***частично рекурсивной функцией*** и, наоборот, если они найдены для всех значений области определения, то её называют ***общерекурсивной функцией***.

Для формирования механизма вычисления рекурсивных функций даны наборы простейших базовых функций: ***константы***, ***тождества*** и ***следования*** и наборы элементарных операций: ***суперпозиции, рекурсии***, ***минимизации*** и ***итерации***.

1.1. Базовые функции

Функция константа. Если $f=\{(x_1, x_2,..., x_n, y)| x_i \in N, y \in N\}=C_n$ или $y = f(x_1, x_2,\ldots x_n)=C_n$, то любым значениям независимых переменных аргумента функции ставится в соответствие значение функции, равное постоянной величине (константе) - C_n, где n –число независимых переменных аргумента. Чаще всего $C_n=0$. Поэтому её называют также ***нуль-функцией***. Например, если $f=\{(x_1, x_2, x_3, y)\}=C_3$, то для $x_1 = 5$, $x_2 = 4$, $x_3 = 7$ и $C_3 = 0$ имеем $y=f(5, 4, 7)=0$ или, если $C_n=1$ при тех же значениях независимых переменных, то $y=f(5, 4, 7)=1$.

Функция тождества. Если $f=\{(x_1, x_2,..., , x_n, y)| x_i \in N, y \in N\}=I_n^m$, то любым значениям независимых переменных аргумента функции ставится в соответствие ее значение, равное значению m-го независимого переменного аргумента, где $1 \le m \le n$ – место независимого переменного аргумента в их упорядоченном списке. Поэтому её называют также ***функцией выбора аргумента***. Например, если $f=\{(x_1, x_2, x_3, y)\}=I_3^2$, то для $x_1 = 5$, $x_2 = 4$, $x_3 = 7$ имеем $y = 4$,

если $f=\{(x_1, x_2, x_3, y)\}=I_3^3$, то для $x_1 = 5$, $x_2 = 4$, $x_3 = 7$ имеем $y = 7$.

Функция следования. Если $f=\{(x, y)| x_i \in N, y \in N\}=\lambda(x)$, то любому значению независимой переменной аргумента ставится в соответствие значение функции, равное числу, непосредственно следующему за числом, являющимся значением независимой переменной. Например,

если для $f=\{(x, y)\}=\lambda(x)$ дано $x = 5$, то $y = (5+1) = 6$,

если для $f=\{(x, y)\}=\lambda(x)$ дано $x = 7$, то $y = (7+1) = 8$.

---

[*] см. [4].

1.2. Элементарные операции

Операции, с помощью которых из простейших базовых функций могут быть получены различные рекурсивные функции, называют ***элементарными.***

Операция суперпозиции. Если даны функция h от m независимых переменных, т.е. $h^{(m)} = (z_1, z_2,\ldots, z_m, y| z_i \in N, y \in N)$, и m функций g_i от n независимых переменных, т.е. $g_i^{(n)} = \{(x_{1i}, x_{2i},..., x_{ni}, z_{ij})| x_{ij} \in N, z_{ji} \in N\}$, то в результате подстановки функций $g_1^{(n)}, g_2^{(n)},\ldots, g_m^{(n)}$ вместо независимых переменных функции $h^{(m)}$ может быть получена новая функция $f^{(n)} = (x_1, x_2,.., x_n, y)$ от n независимых переменных, т.е.

$h^{(m)} = (z_i, z_2,..., z_m, y)$,
$g_1^{(n)} = (x_1, x_2,..., x_n, z_1)$,
$g_2^{(n)} = (x_1, x_2,..., x_n, z_2)$,
.....
$g_m^{(n)} = (x_1, x_2,..., x_n, z_m)$...

$h^{(m)} = (g_1^{(n)}, g_2^{(n)},..., g_m^{(n)}, y)$ или $f^{(n)} = (x_1, x_2,..., x_n, y)$.

Значения функций $g_1^{(n)}, g_2^{(n)},..., g_m^{(n)}$, найденные для данных независимых переменных $x_1 = a_1, x_2 = a_2,..., x_n = a_n$, принять за независимые переменные аргумента функции $h^{(m)}$ и вычислить ее значение, которое принять за значение функции $f^{(n)} = (x_1. x_2,..., x_n, y)$.

Пример 1. Пусть $h^{(1)}=\lambda=(z, y)$ или $y=h^{(1)}(z)=\lambda(z)$ и $g^{(1)}=\lambda=(x, z)$ или $z=g^{(1)}(x)=\lambda(x)$.

Тогда $h^{(1)} = (g^{(1)}(x), y) = \lambda(\lambda(x))$ или $y = h^{(1)}(g^{(1)}(x))= \lambda(\lambda(x))=\lambda^2(x)$.
При $x = 5$ имеем $z = 5+1 = 6$ и $y = 6+1 = 7$.

Пример 2. Пусть $h_*^{(2)} = (z_1, z_2, y)$ или $y = h_*^{(2)}(z_1, z_2) = (z_1 \cdot z_2)$,
$g_{1exp}^{(1)} = (x, z_1)$ или $z_1 = g_{1exp}^{(1)}(x) = x^2$,
$g_{2*}^{(1)} = (x, z_2)$ или $z_2 = g_{2*}^{(1)}(x) = 3 \cdot x$.
Тогда $h_*^{(2)} = (g_{1exp}^{(1)}(x), g_{2*}^{(1)}(x), y) = x^2 \cdot 3 \cdot x$ или $y = 3 \cdot x^3$.
При $x = 2$ имеем $z_1 = 4$, $z_2 = 6$ и $y = 4 \cdot 6 = 3 \cdot 8 = 24$.

Оператор суперпозиции записывают так:

$$f = (x_1, x_2,..., x_n, y) = S_n^m(h^{(m)}, g_1^{(n)}, g_2^{(n)},..., g_m^{(n)}),$$

где $h^{(m)} = (z_1, z_2,..., z_m, y)$ и $g_i^{(n)} = (x_{1i}^{(n)}, x_{2i}^{(n)},..., x_{ni}^{(n)}, z_i)$.

Основные свойства функций, вычисляемых с помощью оператора суперпозиции:

- если вычислимы функции $h^{(m)}$ и $g_i^{(n)}$, то вычислима также функция $f = (x_1, x_2,..., x_n, y) = S_n^m(h^{(m)}, g_1^{(n)}, g_2^{(n)},..., g_m^{(n)})$;
- если даны функции тождества I_n^m и оператор суперпозиции S_n^m, то заданными являются любые операторы подстановки, перестановки и переименования любых независимых переменных аргумента;

• если среди функций g_i имеется хотя бы одна частично рекурсивная, то и функция f также будет частичной.

Пример 3. Перестановка и/или переименование независимых переменных аргумента функции h при n=m: $h^{(m)} = (z_i, z_2,..., z_m, y)$,

$g_1^{(m)} = (x_1, x_2,..., x_m, z_1) = I_m^3$,

$g_2^{(m)} = (x_1, x_2,..., x_m, z_2) = I_m^5$,

.....

$g_m^{(m)} = (x_1, x_2,..., x_m, z_m) = I_m^m$,

$h^{(m)} = (g_1^{(m)}, g_2^{(m)},..., g_m^{(m)}, y)$ или $f^{(m)} = (x_3, x_5,..., x_m, y)$.

Пример 4. Циклическая перестановка независимых переменных аргумента функции h при n = m:

$h^{(m)} = (z_1, z_2,..., z_m, y)$,

$g_1^{(m)} = (x_1, x_2,..., x_m, z_1) = I_m^2$,

$g_2^{(m)} = (x_1, x_2,..., x_m, z_2) = I_m^3$,

.....

$g_{m-1}^{(m)} = (x_1, x_2,..., x_m, z_{m-1}) = I_m^m$,

$g_m^{(m)} = (x_1, x_2,..., x_m, z_m) = I_m^1$,

$h^{(m)} = (g_1^{(m)}, g_2^{(m)},..., g_m^{(m)}, y)$ или $f^{(m)} = (x_2, x_3,..., x_m, x_1, y)$.

Пример 5. Введение дополнительной независимой переменной аргумента:

$h^{(m)} = (z_i, z_2,..., z_m, y)$,

$g_1^{(m+1)} = (x_1, x_2,..., x_{m+1}, z_1) = I_{m+1}^1$,

$g_2^{(m+1)} = (x_1, x_2,..., x_{m+1}, z_2) = I_{m+1}^2$,

.....

$g_m^{(m+1)} = (x_1, x_2,..., x_{m+1}, z_m) = I_{m+1}^m$,

$h^{(m)} = (g_1^{(m+1)}, g_2^{(m+1)},..., g_m^{(m+1)}, y)$ или $f^{(m+1)} = (x_2, x_3,..., x_{m+1}, y)$.

Операция рекурсии. Если даны n-местная рекурсивная функция $g^{(n)}$ и (n+2)-местная рекурсивная функция $h^{(n+2)}$, то можно найти (n+1)-местную рекурсивную функцию $f^{(n+1)}$, используя ***схему примитивной рекурсии***:

$$\begin{cases} f^{n+1}(x_1, x_2,..., x_n, 0) = g^{(n)}(x_1, x_2,..., x_n) = C_n, \\ f^{(n+1)}(x_1, x_2,..., x_n, y+1) = h^{(n+2)}(x_1, x_2,..., x_n, y, f^{(n+1)}(x_1, x_2,..., x_n, y)), \; y \geq 0. \end{cases}$$

Главным дополнительным аргументом функции $h^{(n+2)}$ является y, который указывает, при каком его значении следует определять значение функции $f^{(n+1)}(x_1, x_2,\ldots, x_n, y)$ для вычисления последующих значений функции $f^{(n+1)}(x_1, x_2,\ldots, x_n, y+1)$.

Оператор рекурсии описывают так:

$f^{n+1}=(x_1, x_2,..., x_n, y+1)=R(g^{(n)}, h^{(n+2)})$,

где $g_i^{(n)}=(x_1, x_2,..., x_n)$ и $h^{(n+2)}=(x_1, x_2,..., x_n, y, f^{(n+1)}(x_1, x_2,..., x_n, y))$.

При исполнении операции рекурсии известными являются $x_1=a_1$, $x_2=a_2$,..., $x_n=a_n$. Значением функции $f^{(n+1)}$ для нулевого значения главного дополнительного аргумента (y=0) есть значение функции $g^{(n)}$. Значением функции $f^{(n+1)}$ для каждого последующего значения главного аргумента (y+1) следует считать значение функции h, вычисленное

по значению главного аргумента y и значению вспомогательного аргумента $f^{(n+1)}(x_1, x_2,..., x_n, y)$. Функции, для вычисления значений которых использовали базовые функции и операторы суперпозиции и примитивной рекурсии, называют ***примитивно рекурсивными функциями***.

При задании примитивно рекурсивного описания функции f(x), зависящей от одной независимой переменной, схема примитивной рекурсии имеет вид: $\begin{cases} f(0) = C_1, \\ f(y+1) = h(y, f(y)),\ y > 0. \end{cases}$

Пример 6. Вычислить ***функцию предшествования***

$$\lambda^{-1}(y) = (y \dot{-} 1) = \begin{cases} y-1, & \text{если } y > 1, \\ 0, & \text{если } y \le 1, \end{cases}$$

где символ «$\dot{-}$» означает оператор усеченного вычитания.

Для этого используем базовую функцию константы: g(0) = C(0) = 0 и базовую функцию тождества: $h(y, f(0, y)) = I_2^1 = y$, т. е.

$$\begin{cases} f(0) = g(0) = 0, \\ f(0, y+1) = h(y, f(0, y)) = y. \end{cases}$$

Так как функции g и h не содержат независимой переменной x, то:

f(0) = 0,

f(1) = 0,

f(2) = 1,

……..

f (i) = (i$\dot{-}$1).

Если i = y, то $f(0, y) = (y \dot{-} 1) = \lambda^{-1}(y)$. Пусть y = 6, тогда $f(6) = \lambda^{-1}(6) = 6 \dot{-} 1 = 5$.

Так можно вычислять рекурсивную функцию предшествования

$\lambda^{-1}(y) = (y \dot{-} 1) = R(0, y)$.

Пример 7. Вычислить функцию f (n) = n.

Для этого используем базовые функции $g(0)=C_1=0$ и $h(0, y, f(0, y))= \lambda(J_{3,2})= =\lambda(y)=y+1$.

По схеме примитивной рекурсии:

f(0)=0,

f(0+1)=0+1=1,
f(1+1)=1+1=2,
f(2+1)=2+1=3,
……
f(i+1)=(i+1).

Следовательно, если i=n, то f(n)=(..((0+1)+1)…+1)=n есть примитивно рекурсивная функция, для вычисления которой по схеме примитивной рекурсии использованы нуль-функция, функция тождества и функция следования.

Пример 8. Вычислить функцию f(x) = x + n.

Если $g(x, 0) = J_{2,1} = x$ и $h(x, y, f(x, y))=\lambda(J_{3,2})=\lambda(y)=y+1$, то по схеме примитивной рекурсии:

f(x, 0)=x,
f(x, 0+1)=x+1,
f(x, 1+1)=(x+1)+1=x+2,
f(x, 2+1)=(x+2)+1=x+3,
……
f(x, i+1)=(x+i)+1=x+(i+1).

Следовательно, если i=n, то f(n)=(…((x+1)+1)…+1)=(x+n) есть примитивно рекурсивная функция, для вычисления которой по схеме примитивной рекурсии использованы базовые функции тождества и следования.

Пример 9. Вычислить ***функцию сложения*** $f_+(x, y) = (x+y)$.

Для этого используем базовые функции тождества $g(x) = I_1^1 = x$ и следования $h(x, y, f_+(x, y))=\lambda(J_{3,3}) = f_+(x, y) + 1$, т. е.

$$\begin{cases} f_+(x, 0) = g(x) = J_{1,1} = x, \\ f_+(x, y + 1) = h(x, y, f_+(x, y)) = \lambda(J_{3,3}) = f_+(x, y) + 1. \end{cases}$$

По схеме примитивной рекурсии имеем:

$f_+(x, 0) = g(x)=x$,
$f_+(x, 1) = h(x, 0, f_+(x, 0)) = x + 1$,
$f_+(x, 2) = h(x, 1, f_+x, 1)) = x + 2$,
$f_+(x, 3) = h(x, 2, f_+(x, 2)) = x + 3$,
……
$f_+(x, i) = h(x, (i-1), f_+(x, (i -1))) = x + i$.

Если i = y, то $f_+(x, y) = (x+y)$. Пусть x=3, y=6, тогда $f_+(3, 6) = 3+6 = 9$.

Так, используя базовые функции тождества и следования, можно вычислять значения функции сложения $f_+(x, y) = (x+y) = R(x, (f(x, y)+1))$.

Функция $f_+(x, y) = R(x, (f_+(x, y)+1)) = (x+y)$ является примитивно рекурсивной.

Пример 10. Вычислить ***функцию усеченного вычитания***

$$f_{\dot{-}}(x, y) = (x \dot{-} y) = \begin{cases} x - y, & \text{если} \quad x > y, \\ 0, & \text{если} \quad x \le y. \end{cases}$$

Для этого используем базовую функцию тождества g(x) = x и примитивно-рекурсивную функцию предшествования $\lambda^{-1}(f(x, y)) = (f(x, y)\dot{-}1)$, т. е.

$$\begin{cases} f_{\square}(x,0) = g(x) = x, \\ f_{\square}(x,y+1) = h(x,y,f_{\square}(x,y)) = \lambda^{-1}(f_{\square}(x,y) = (f_{\square}(x,y)\dot{-}1). \end{cases}$$

По схеме примитивной рекурсии имеем:

$f_{\square}(x, 0) = g(x) = x,$

$f_{\square}(x, 1) = h(x, 0, f_{\square}(x, 0)) = x\dot{-}1,$

$f_{\square}(x, 2) = h(x, 1, f_{\square}(x, 1)) = (x\dot{-}1)\dot{-}1 = x\dot{-}2,$

.....

$f_{\square}(x, i) = h(x, i, f_{\square}(x, i)) = (x\dot{-}(i-1))\dot{-}1 = x\dot{-}i.$

Если i=y, то $f_{\square}(x, y) = x\dot{-}y$. Пусть x = 6, y = 3, тогда $f_{\square}(6, 3) = 6\dot{-}3 = 3$.

Так, используя базовую функцию тождества и примитивно рекурсивную функцию предшествования, можно вычислять значение функции усеченного вычитания $f_{\square}(x,y) = R(x,\lambda^{-1}(f_{\square}(x,y))) = (x\dot{-}y)$.

Пример 11. Вычислить ***функцию умножения*** $f_*(x, y) = x \cdot y$.

Для этого используем базовую нуль-функцию g(x) = C_1 = 0 и примитивно-рекурсивную функцию сложения $h(x, y, f_*(x, y)) = f_+(I_{3,1}, I_{3,3}) = x + f_*(x, y)$, т.е.

$$\begin{cases} f_*(x,0) = g(x) = 0, \\ f_*(x,y+1) = h(x, y, f_*(x,y)) = f_+(I_{3,1}, I_{3,3}) = x + f_*(x,y). \end{cases}$$

По схеме примитивной рекурсии имеем:

$f_*(x,0) = g(x) = 0,$

$f_*(x,1) = h(x, 0, f_*(x,0)) = x+0 = x,$

$f_*(x,2) = h(x, 1, f_*(x,1)) = (x+x) = 2\cdot x,$

$f_\times(x,3) = h(x, 2, f_*(x,2)) = (x+2\cdot x) = 3\cdot x,$

……

$f_*(x,i) = h(x, (i - 1), f_*(x,i-1)) = (x+(i - 1)\cdot x) = i\cdot x.$

Если i = y, то $f_\times(x, y) = x\cdot y$.

Пусть x=3, y=4. Тогда $f_*(3, 4) = 3\cdot 4 = 12$.

Так, используя базовую функцию константы и примитивно рекурсивную функцию сложения, можно вычислять значения функции умножения.

Функция $f_*(x, y) = R(0, f_+(x, f_*(x, y))) = (x\cdot y)$ является рекурсивной.

Пример 12. Вычислить функцию f(n)=n!

Если g(x, 0) = 1 и $h(x, y, f(x, y)) = f_\times(\lambda(J_{3,2}), J_{3,3})$, то по схеме примитивной рекурсии:

f(x, 0)=1,

f(x, 0+1)=1·1=1,

f(x, 1+1)=2·1=2,
f(x, 2+1)=3·2=6,
f(x, 3+1)=4·6=24
……
f(i+1)=(i+1)·i·(i-1)·(i-2)·…·(i-i)=(i +1)!

Если (i+1)=n, то f(n)=n! есть рекурсивная функция, так как для ее вычисления использованы базовые функции константы, тождества и следования и рекурсивная функция умножения.

Пример 13. Вычислить ***функцию возведения в степень*** $f_{exp}(x, y) = x^y$.

Для этого используем базовую функцию константы $g(x)=C_1=1$ и рекурсивную функцию умножения $h(x, y, f_{exp}(x, y))= f_{\times}(I_{3,1}, I_{3,3})=x\cdot f_{exp}(x, y)$:

$$\begin{cases} f_{exp}(x, 0) = g(x) = 1, \\ f_{exp}(x, y+1) = h(x, y, f_{exp}(x, y)) = x\cdot f_{exp}(x, y). \end{cases}$$

По схеме примитивной рекурсии имеем:
$f_{exp}(x, 0)=g(x)=1$;
$f_{exp}(x, 1)=h(x, 0, f_{exp}(x, 0))=x\cdot 1= x$;
$f_{exp}(x, 2)=h(x, 1, f_{exp}(x, l))=x\cdot x=x^2$;
$f_{exp}(x, 3)=h(x, 2, f_{exp}(x, 2))=x\cdot x^2=x^3$,
……..
$f_{exp}(x, i)=h(x, (i - 1), f_{exp}(x, (i - 1)))=x\cdot x^{(i - 1)} = x^i$.
Если i=y, то $f_{exp}(x, y)=x^y$.
Пусть x=3, y=3, тогда $f_{exp}(3, 3)=3^3 =27$.

Так, используя базовую функцию C_1 и рекурсивную функцию умножения, можно вычислять значения рекурсивной функции возведения в степень
$x^y =R(1, f_{\times}(x, f_{exp}(x, y)))=f_{exp}(x, y)$.

Пример 14. Вычислить ***функцию*** $Sg(y) = \begin{cases} 0, \text{ если } y = 0, \\ 1, \text{ если } y > 0, \end{cases}$

где $Sg^*(y)$ – знак переменной y.
Пусть $g(x) = C_1 = 0$ и $h(x, y, Sg(x, y)) = C_3 = 1$, т.е.

$$\begin{cases} Sg(x, 0) = g(x) = 0, \\ Sg(x, y+1) = h(x, y, Sg(x, y)) = 1. \end{cases}$$

По схеме примитивной рекурсии имеем:
Sg(x, 0) = g(x) = 0,
Sg(x, 1) = h(x, 0, Sg(x, 0)) = 1,
Sg(x, 2) = h(x, 1, Sg(x, 1)) = 1,
Sg(x, 3) = h(x, 2, Sg(x, 2)) = 1,
……
Sg(x, i) = h(x, (i-1), Sg(x, (i-1))) = 1.

* лат. signum – знак.

Используя две базовых функции константы, можно вычислять знаковую примитивно рекурсивную функцию Sg(y) = 1 для y>0.

Пример 15. Вычислить ***функцию*** $\overline{Sg}(y) = 1 - Sg(y) = \begin{cases} 1, \text{ если } y = 0, \\ 0, \text{ если } y > 0, \end{cases}$

где $\overline{Sg}(y)$ - инверсия знака y.

Пусть g(x) = C_1 = 1 и h(x, y, $\overline{S}g(x, y)$) = C_3 = 0, т.е.

$$\begin{cases} \overline{S}g(x, 0) = g(x) = 1, \\ \overline{S}g(x, y+1) = h(x, y, \overline{S}g(x, y). \end{cases}$$

По схеме примитивной рекурсии имеем:

$\overline{S}g(x, 0) = g(x) = 1,$

$\overline{S}g(x, 1) = h(x, 0, \overline{S}g(x, 0)) = 0,$

$\overline{S}g(x, 2) = h(x, 1, \overline{S}g(x, 1)) = 0,$

$\overline{S}g(x, 3) = h(x, 2, \overline{S}g(x, 2)) = 0,$

……

$\overline{S}g(x, i) = h(x, (i-1), \overline{S}g(x, (i-1))) = 0.$

Используя две базовых функции константы, можно вычислять знаковую примитивно рекурсивную функцию $\overline{Sg}(y) = 0$ для y>0.

Пример 16. Вычислить min{x, y}.

Эту функцию можно вычислить с помощью примитивно рекурсивной функции «$\dot{-}$» или «Sg»:

$\min\{x, y\} = x \dot{-} (x \dot{-} y)$ или $\min\{x, y\} = x \cdot Sg(y \dot{-} x) + y \cdot \overline{Sg}(y \dot{-} x).$

Пусть x = 5, y = 3.

Тогда $\min\{5, 3\} = 5 \dot{-} (5 \dot{-} 3) = 3$ или $\min\{5,3\} = 5 \cdot Sg(3 \dot{-} 5) + 3 \cdot \overline{Sg}(3 \dot{-} 5) = 3.$

Пусть x = 3, y = 5.

Тогда $\min\{3, 5\} = 3 \dot{-} (3 \dot{-} 5) = 3$ или $\min\{3,5\} = 3 \cdot Sg(5 \dot{-} 3) + 5 \cdot \overline{Sg}(5 \dot{-} 3) = 3.$

Функция min{x, y} является рекурсивной.

Пример 17. Вычислить max{x, y}.

Функцию также можно вычислить с помощью примитивно рекурсивной функции «$\dot{-}$» или «Sg»:

$\max\{x, y\} = y + (x \dot{-} y)$ или $\max\{x, y\} = x \cdot Sg(x \dot{-} y) + y \cdot \overline{Sg}(x \dot{-} y).$

Пусть x = 5, y = 3.

Тогда $\max\{5, 3\} = 3 + (5 \dot{-} 3) = 5$ или $\max\{5, 3\} = 3 + (5 \dot{-} 3) = 5.$

Пусть x =3, y = 5.

Тогда $\max\{3, 5\} = 5 + (3 \dot{-} 5) = 5$ или $\max\{3, 5\} = 3 \cdot Sg(3 \dot{-} 5) + 5 \cdot \overline{Sg}(3 \dot{-} 5) = 5.$

Функция max{x, y} является рекурсивной.

Пример 18. Вычислить $|x \dot{-} y|$.

Эту функцию можно вычислить с помощью одной примитивно рекурсивной функции «$\dot{-}$»:

$|x \dot{-} y| = (x \dot{-} y) + (y \dot{-} x).|$

Пусть x = 5, y = 3. Тогда $|5 \dot{-} 3| = (5 \dot{-} 3) + (3 \dot{-} 5) = 2+0=2$.

Пусть x = 3, y = 5. Тогда $|3 \dot{-} 5| = (3 \dot{-} 5) + (5 \dot{-} 3) = 0+2=2$.

Функция $|x \dot{-} y|$ является рекурсивной.

Пример 19. Вычислить логические функции $(x \vee y)$, $(x \& y)$, $\neg x$.

С помощью примитивно рекурсивной функции усеченного вычитания возможна «арифметизация» логических функций, аргументы и значения которых принадлежат множеству {0, 1}:

a) $(x \vee y) = \max(x, y) = y + (x \dot{-} y)$,

b) $(x \& y) = \min(x, y) = x \dot{-} (x \dot{-} y)$,

c) $\neg x = 1 \dot{-} x$.

Пусть x = 0, y = 0. Тогда $(0 \vee 0) = 0 + (0 \dot{-} 0) = 0$, $(0 \& 0) = 0 \dot{-} (0 \dot{-} 0) = 0$, $\neg x = \neg 0 = 1 \dot{-} 0 = 1$.

Пусть x = 1, y = 0. Тогда $(1 \vee 0) = 0 + (1 \dot{-} 0) = 1$, $(1 \& 0) = 1 \dot{-} (1 \dot{-} 0) = 0$, $\neg x = \neg 1 = 1 \dot{-} 1 = 0$.

Пусть x =0, y = 1. Тогда $(0 \vee 1) = 1 + (0 \dot{-} 1) = 1$, $(0 \& 1) = 0 \dot{-} (0 \dot{-} 1) = 0$,

Пусть x = 1, y = 1. Тогда $(1 \vee 0) = 1 + (1 \dot{-} 1) = 1$, $(1 \& 0) = 1 \dot{-} (1 \dot{-} 1) = 1$.

Множество логических операторов $\{\neg, \&, \vee\}$ представляют функционально полную систему. Из этого следует, что любые логические функции являются рекурсивными.

Пример 20. Вычислить предикат $P(x_1, x_2,\ldots, x_n)$.

Для «арифметизации» вычисления предикатов $P(x_1, x_2,\ldots, x_n)$, которые принимают значения на множестве {истина, ложь}, следует ввести характеристическую функцию $\chi_p(x_1, x_2,..., x_n)$, значения которой есть

$$\chi_p(x_1, x_2,..., x_n) = \begin{cases} 1, \text{ если } P(x_1, x_2,..., x_n) = \text{и}, \\ 0, \text{ если } P(x_1, x_2,..., x_n) = \text{л}. \end{cases}$$

Очевидно, что $\chi_p(x_1, x_2,..., x_n) = Sg(x_1, x_2,\ldots, x_n)$. Следовательно, если характеристическая функция примитивно рекурсивная, то предикат также примитивно рекурсивный. Если предикаты $P_1, P_2,\ldots, P_n$ примитивно рекурсивные, то любой предикат, полученный из $P_1, P_2,\ldots, P_n$ с помощью логических связок, также примитивно рекурсивен.

Пример 21. Вычислить отношение между объектами $r(x_1, x_2,\ldots, x_n)$.

Между математическими объектами могут быть установлены отношения

$\theta=\{=, <, \le, \ldots\}$, т.е. $r_{=}(x_1, x_2, \ldots, x_n)$, $r_{<}(x_1, x_2, \ldots, x_n)$, $r_{\le}(x_1, x_2, \ldots, x_n)$ и др.
Для «арифметизации» таких отношений следует также ввести характеристическую функцию:

$$\chi_p(x_1,\ x_2,...,\ x_n) = \begin{cases} 1, \text{ если } r(x_1, x_2,..., x_n) = \text{и}, \\ 0, \text{ если } r(x_1, x_2,..., x_n) = \text{л}. \end{cases}$$

Пример 22. Вычислить функцию деления x на y.

С помощью примитивно рекурсивных функций «$\dot{-}$» и «$\overline{Sg}$» можно найти рекурсивное описание функции деления:

$f(x, y) = \left[\frac{x}{y}\right]$ - целая часть деления x на y:

$$\left[\frac{x}{y}\right] = \sum_{i=1}^{x} \overline{Sg}(i \cdot y \dot{-} x).$$

$f(x, y) = rest(x, y)$ – остаток от деления x на y:

$$rest(x, y) = x \dot{-} y \cdot \left[\frac{x}{y}\right].$$

Пусть x=8, y=3. Очевидно, $\frac{8}{3} = 2\frac{2}{3}$, т.е. целая часть и остаток равны 2.

$$\left[\frac{8}{3}\right] = \overline{Sg}(1 \cdot 3 \dot{-} 8) + \overline{Sg}(2 \cdot 3 \dot{-} 8) + \overline{Sg}(3 \cdot 3 \dot{-} 8) = 1 + 1 + 0 = 2,$$

$rest(8, 3) = 8 \dot{-} 3 \cdot 2 = 8 \dot{-} 6 = 2$. *ч.т.д.*[*]

Пусть x = 3, y = 8. Очевидно, что $\frac{3}{8} = 0\frac{3}{8}$, т.е. целая часть равна нулю, а остаток - трем.

$$\left[\frac{3}{8}\right] = \overline{Sg}(1 \cdot 8 \dot{-} 3) + \overline{Sg}(2 \cdot 8 \dot{-} 3) + \overline{Sg}(3 \cdot 8 \dot{-} 3)... = 0 + 0 + 0... = 0,$$

$rest(3, 8) = 3 \dot{-} 8 \cdot 0 = 3 \dot{-} 0 = 3$. *ч.т.д.*

Пример 23. Вычислить $\sum_{y=0}^{n} f(x_1, x_2,..., x_n, y)$.

Пусть $g(x_1, x_2,..., x_n, 0) = 0$. Тогда

$f_+^{(n+1)}(x_1, x_2,..., x_n, y+1) = h^{(n+2)}((x_1, x_1,..., x_n), y, f_+^{(n+1)}(x_1, x_1,..., x_n, y)) = f_+(I_{3,2}, I_{3,3}) = f_+(y,\ f_+^{(n+1)}(x_1, x_2,..., x_n, y))$.

То есть $f_+^{(n+1)}(x_1, x_2,..., x_n, n) = \sum_{y=0}^{n} f(x_1, x_2,..., x_n, y)$.

[*] *ч.т.д.*- что требовалось доказать.

Рекурсивная функция $\sum_{y=0}^{n} f(x_1, x_2, ..., x_n, y)$ может быть вычислена по схеме примитивной рекурсии, используя примитивно рекурсивную функцию сложения.

Пример 24. Вычислить $f(x, y) = x + \sum_{i=0}^{i=y} i$.

Пусть $g(x) = J_{1,1}=x$ и $h(x, y, f(x, y)) = f_+(J_{3,2}, J_{3,3})=y+f(x,y)$.

По схеме примитивной рекурсии:
$f(x, 0)=g(x)=x$,
$f(x, (0+1))=0+x=x$,
$f(x, (1+1))=1+x$,
$f(x, (2+1))=2+1+x=3+x$,
$f(x, (3+1))=3+3+x=6+x$,
$f(x, (4+1))=4+6+x=10+x$,
$f(x, (5+1))=5+10+x=15+x$,
$f(x, (6+1))=6+15+x=21+x$,
$f(x, (7+1))=7+21+x=28+x$,
……
$f(x, (i+1))=x+\sum_{i=0}^{i=y} i$.

Следовательно, $f(x, y) = x + \sum_{i=0}^{i=y} i$ есть рекурсивная функция, для вычисления которой привлечены базовая функция тождества и примитивно рекурсивная функция суммирования.

Пример 25. Вычислить $\prod_{y=0}^{n} f(x_1, x_2, ..., x_n, y)$.

Пусть $g(x_1, x_2, ..., x_n, 0) = 1$. Тогда

$f_*^{(n+1)}(x_1, x_2, ..., x_n, y+1) = h^{(n+2)}((x_1, x_1, ..., x_n), y, f_*^{(n+1)}(x_1, x_1, ..., x_n, y)) =$

$f_*(I_{3,2}, I_{3,3}) = f_*(y, f_*^{(n+1)}(x_1, x_2, ..., x_n, y))$.

То есть $f_*(x_1, x_2, ..., x_n, n) = \prod_{y=0}^{n} f(x_1, x_2, ..., x_n, y)$.

Рекурсивная функция $\prod_{y=0}^{n} f(x_1, x_2, ..., x_n, y)$ может быть вычислена по схеме примитивной рекурсии, используя примитивно рекурсивную функцию умножения.

Таким образом, из элементарных функций C_n, I_n^m и $\lambda(x+1)$ с помощью операторов суперпозиции и примитивной рекурсии были получены основные функции арифметики, алгебры и анализа. Тем самым было показано, что эти

функции имеют примитивно рекурсивное описание, которое однозначно определяет процедуру их вычисления.

Последовательность рекурсивного описания, использующая базовые функции и результаты вычисления функции на всех предшествующих точках с помощью операторов суперпозиции и примитивной рекурсии, называют протоколом.

Операция минимизации (или **поиск наименьшего корня).** Если дана (n+1)-местная функция $f(x_1, x_2,..., x_n, y)$, то n-местную функцию $\varphi(x_1, x_2,..., x_n)$ можно вычислить при заданных $x_1 = a_1$, $x_2 = a_2$,..., $x_n = a_n$, придавая вспомогательному аргументу y последовательно значения 0, 1, 2, ..., пока $f(a_1, a_2,..., a_n, y)$ не окажется в первый раз (!) равной нулю, т.е. $f(a_1, a_2,..., a_n, y) = 0$. Полученное значение y принять за значение определяемой функции, т.е. $y = \varphi(a_1, a_2,..., a_n)$. Поиск значений функции $\varphi(x_1, x_2,..., x_n)$ выполняется с помощью μ - оператора:

$$\varphi(x_1, x_2,..., x_n)=\mu_y(f(x_1, x_2,..., x_n, y)=0).$$

Алгоритм вычисления функции $\varphi(a_1, a_2,..., a_n)$:
шаг 1: принять y=0 и вычислить функцию $f(a_1, a_2,..., a_n, y)$.
шаг 2: если $f(a_1, a_2,..., a_n, y)=0$, то конец, значение функции $\varphi(a_1, a_2,..., a_n)=y$, иначе принять y=y+1 и перейти к шагу 1.

Пример 26. Пусть $f(a_1, a_2,..., a_n, y)=(x_1 - x_2 - x_3 - y)$ для $x_1=7$, $x_2=1$, $x_3=2$.
Найти $\varphi(x)$.
По данному алгоритму имеем:
$f(7, 1, 2, 0) = 7-1-2-0 \neq 0$,
$f(7, 1, 2, 1) = 7-1-2-1 \neq 0$,
$f(7, 1, 2, 2) = 7-1-2-2 \neq 0$,
$f(7, 1, 2, 3) = 7-1-2-3 \neq 0$,
$f(7, 1, 2, 4) = 7-1-2-4 = 0$.
Следовательно, $\varphi(x_1, x_2,..., x_n) = \varphi(7, 1, 2)=4$.

Пример 27. Дана функция $f(x_1, x_2, y) =(x_1 \cdot y - x_2)$ при $x_1 = 3$ и $x_2 = 9$. Вычислить значение функции $\varphi(x_1, x_2)$.

Так как $\varphi(x_1, x_2) = \mu_y(f(x_1, x_2, y) = 0) = (x_1 \cdot y - x_2)= 0$, то $\varphi(x_1, x_2) = y = \left[\frac{x_2}{x_1}\right]$.

Если $\varphi(x_1, x_2)$ - рекурсивная функция, то $\left[\frac{x_2}{x_1}\right]$ должна иметь значения x_2 кратные x_1, т.е. $\varphi(x_1, x_2)=\left[\frac{x_2}{x_1}\right]= 1, 2, 3, \ldots$

Например, если $x_1=3$, то $\varphi(3, x_2)$ определена на множестве {3, 6, 9, 12, ...} и имеет значение только на множестве целых чисел 1, 2, 3, 4,..., соответственно.

Таким образом, функция $\varphi(x_1, x_2)$ является частично рекурсивной. Если $x_1 = 3$, $x_2 = 9$, то $y = \varphi(3, 9) = 3$.

Пример 28. Дана $f(x, y)=y^2 - x$. Найти $\varphi(x)$.

Так как $\varphi(x) = \mu_н(f(x, y) = 0) = (y^2 - x)$, то $\varphi(x) = y = \left[\sqrt{x}\right]$. Если $\varphi(x)$ является рекурсивной функцией, то $\left[\sqrt{x}\right]$ должна принимать значения только на множестве целых чисел 1, 2, 3,... Следовательно, функция $\varphi(x)$ определена на множестве {1, 4, 9, 16,…}.

Пример 29. Дана $f(x, y)=\left[x - \frac{y}{2}\right]$. Найти $\varphi(x)$.

Для $x = 3$ имеем $f(3, 0) \neq 0$, $f(3, 1) \neq 0$, $f(3, 2) \neq 0$, $f(3, 3) \neq 0$, $f(3, 4) \neq 0$, $f(3, 5) \neq 0$, $f(3, 6) = 0$, т.е $\varphi(3) = 6$.

Операция итерации. Многократное повторение данного процесса, пока не будет выполнено некоторое условие, называют **итерацией**. При этом на каждом шаге итерации данный процесс выполняется полностью. Условием итерации, как правило, является число повторений. Оператор итерации, формирующий новую функцию, обозначается так $f(n) = J^n(h)$, где h – функция, формирующая новую функцию $f(i+1)$ итеративной процедурой для $1 \leq i \leq n$, начальное значение которой $f(0) = 0$:

$$\begin{cases} f(0) = 0, \\ f(i+1) = h(f(i)). \end{cases}$$

Пример 30. Дана функция $h(f(i)) = (1 + \left[\frac{f(i)}{2}\right])$. Какую функцию вычисляет оператор $J^n(h)$?

$f(0) = 0,$

$$f(1) = (h(f(0)) = (1 + \left[\frac{0}{2}\right])) = 1,$$

$$f(2) = (h(f(1)) = (1 + \left[\frac{1}{2}\right])) = 1,$$

$$f(3) = (h(f(2)) = (1 + \left[\frac{1}{2}\right])) = 1,$$

……..

$$f(i) = (h(f(i-1)) = (1 + \left[\frac{1}{2}\right])) = 1.$$

Если i=n, то $f(n) = (h(f(n-1)) = (1 + \left[\frac{1}{2}\right])) = 1.$

Следовательно, оператор $J^n(h)$ вычисляет рекурсивную функцию $f(n) = Sg(n)$, где $1 \le i \le n$.

Пример 31. Дана функция $h(f(i)) = (f(i) + 1 + \left[\frac{f(i)+1}{2}\right] - \left[\frac{f(i)}{2}\right])$.

Какую функцию вычисляет оператор $J^n(h)$?

$f(0) = 0,$

$$f(1) = (h(f(0)) = (0 + 1 + \left[\frac{0+1}{2}\right] - \left[\frac{0}{2}\right])) = 1,$$

$$f(2) = (h(f(1)) = (1 + 1 + \left[\frac{1+1}{2}\right] - \left[\frac{1}{2}\right])) = 3,$$

$$f(3) = (h(f(2)) = (3 + 1 + \left[\frac{3+1}{2}\right] - \left[\frac{3}{2}\right])) = 5,$$

……

$$f(i) = (h(f(i-1)) = (f(i-1) + 1 + \left[\frac{f(i-1)+1}{2}\right] - \left[\frac{f(i-1)}{2}\right])) = (2 \cdot i - 1).$$

Если i=n, то $f(n) = (2 \cdot n - 1)$, т.е. оператор $J^n(h)$ вычисляет рекурсивную функцию $f(n) = (2 \cdot n - 1)$.

Пример 32. Дана функция $h(f(i)) = (a \cdot f(i)) + b)$. Какую функцию вычисляет оператор $J^n(h)$?

$f(0) = 0,$

$$f(1) = (h(f(0)) = (a \cdot 0 + b)) = b,$$

$$f(2) = (h(f(1)) = (a \cdot b + b)) = b \cdot (a + 1),$$

$$f(3) = (h(f(2)) = (a \cdot (a \cdot b + b) + b)) = b \cdot (a^2 + a + 1),$$

$$f(4) = (h(f(3)) = (a \cdot b \cdot (a^2 + a + 1) + b)) = b \cdot (a^3 + a^2 + a + 1),$$

……

$$f(i) = (h(f(i-1)) = b \cdot (a^{i-1} + a^{i-2} + a^{i-(i-1)} + 1).$$

Если i=n, то $f(n) = b \cdot (a^{n-1} + a^{n-2} + \ldots + a + 1)$, т. е. оператор $J^n(h)$ вычисляет рекурсивную функцию.

Итак, функция называется ***примитивно рекурсивной***, если она получена из базовых функций с помощью конечного числа операторов суперпозиции и/или примитивной рекурсии. Она же называется ***рекурсивной***, если получена с помощью примитивно рекурсивных функций и конечного числа операторов суперпозиции, примитивной рекурсии и/или минимизации. Иначе говорят так: функции, для которых существуют алгоритмы вычисления, называют ***рекурсивными функциями.*** Рекурсивная функция называется частично рекурсивной, если она определена не на всем множестве целых положительных чисел и общерекурсивной, если она определена на всем множестве целых положительных чисел.

1.3. Нумерация наборов чисел

В теории алгоритмов получил распространение приём, позволяющий сводить изучение функции от нескольких независимых переменных аргумента к изучению функции от одной переменной. Знание этого метода необходимо для организации использования, распределения и хранения данных в памяти компьютера. Этот метод основан на нумерации наборов чисел так, что существует однозначное отображение h(x, y) каждого набора соответствующему номеру $n \in N$. При обратном отображении по номеру можно восстановить набор независимых переменных. Например, для функции, содержащей две независимых переменных (x, y), это отображение h(x, y) может быть таким:

(x, y)	(0, 0)	(0, 1)	(1, 0)	(0, 2)	(1, 1)	(2, 0)	(0, 3)	(1, 2)	(2, 1)	(3, 0)	…
h (x, y)	0	1	2	3	4	5	6	7	8	9	…

Пусть пары (x, y) формируют частично упорядоченное множество $N^{(2)}$.

Если дано h(x, y) = n, то существует обратное отображение: $x = h_1^{-1}(n)$ и $y = h_2^{-1}(n)$, т. е. $h(h_1^{-1}(n), h_2^{-1}(n)) = n$.

Это позволяет вычислять номер n для любой пары (x, y) и, наоборот, по номеру n вычислять значения x и y:

$$h(x,y) = \left[\frac{(x+y)\cdot(x+y+1)}{2} + x\right] = n;$$

$$x = h_1^{-1}(n) = n - \frac{1}{2}\left[\frac{\left[\sqrt{8n+1}\right]+1}{2}\right]\cdot\left[\frac{\left[\sqrt{8n+1}\right]-1}{2}\right];$$

$$y = h_2^{-1}(n) = \left[\frac{\left[\sqrt{8n+1}\right]+1}{2}\right] + \frac{1}{2}\left[\frac{\left[\sqrt{8n+1}\right]+1}{2}\right]\cdot\left[\frac{\left[\sqrt{8n+1}\right]-1}{2}\right] - n - 1. \quad (1)$$

Используя эти правила, можно вычислять нумерацию троек $h^2(x, y, z) = h(h(x, y), z) = n$ и, наоборот, по номеру тройки - значения x, y, z.

Например, если $h^2(x, y, z) = n$, то

$$z = h_2^{-1}(n),$$

$$y = h_2^{-1}(h_1^{-1}(n)), \quad (2)$$

$$x = h_1^{-1}(h_1^{-1}(n)),$$

$$h^2(x, y, z) = h(h(h_1^{-1}(h_1^{-1}(n)), h_2^{-1}(h_1^{-1}(n))), h_2^{-1}(n)).$$

Тройки (x, y, z) формируют частично упорядоченное множество $N^{(3)}$.

Аналогично для произвольного количества чисел имеем:

$h^{n-1}(x_1, x_2,\ldots, x_n)=h(h\ldots h(h(x_1, x_2), x_3)\ldots x_{n-1}), x_n)$.

Если $h^{n-1}(x_1, x_2,\ldots, x_n)=m$, то

$$x_n = h_2^{-1}(m),$$
$$x_{n-1} = h_2^{-1}(h_1^{-1}(m)),$$
$$\text{.....}, \qquad (3)$$
$$x_2 = h_2^{-1}(h_1^{-1}(...h_1^{-1}(m)...)),$$
$$x_1 = h_1^{-1}(h_1^{-1}(...h_1^{-1}(m)...)).$$
$$x_n = h_2^{-1}(m),$$
$$x_{n-1} = h_2^{-1}(h_1^{-1}(m)),$$
$$\text{.....}, \qquad (3)$$
$$x_2 = h_2^{-1}(h_1^{-1}(...h_1^{-1}(m)...)),$$
$$x_1 = h_1^{-1}(h_1^{-1}(...h_1^{-1}(m)...)).$$

Имея нумерацию множеств наборов $N^{(1)}, N^{(2)},..., N^{(i)},..., N^n$, где $N^{(i)}$ – множество наборов (i) чисел, можно установить объединенную нумерацию произвольных наборов чисел $M = N^{(1)} \cup N^{(2)} \cup ... \cup N^{(i)} \cup ... \cup N^{(n)}$, где $M \subseteq N$.

Для любого $n \in N$ имеем $h(x_1, x_2,..., x_n) = h(h^{n-1}(x_1, x_2,..., x_n), n-1)$.

Если $h(x_1, x_2,..., x_n) = m$, то $h^{n-1}(x_1, x_2,..., x_n) = h_1^{-1}(m)$, (4)

$n = h_2^{-1}(m) + 1$.

Используя (3), можно восстановить значения $x_1, x_2, \ldots, x_n$.

2. Машина Тьюринга - модель алгоритма

Основные свойства алгоритма дискретности, детерминизма, массовости и результативности позволяют представить процесс вычисления какой-либо числовой функции с помощью математической машины. Эта машина за конечное число шагов позволяет вычислить по исходным данным искомый числовой результат в соответствии с заданными правилами.

Такая модель алгоритма была предложена английским математиком Тьюрингом в конце 30-х годов прошлого столетия, что почти на два десятилетия опередило появление электронных вычислительных машин и послужило их теоретическим прообразом. Машина Тьюринга состоит из информационной ленты, считывающей и записывающей головки и управляющего устройства (рис. 1).

Информационная лента бесконечной длины представляет собой последовательность ячеек, в каждую из которых записан в точности только один символ из множества символов алфавита $V_T=\{a_1, a_2,...,a_n\}$. Последовательность символов на ленте формирует слово $\alpha=(a_1\ a_2,...,\ a_|)$. Пробел между словами также является символом множества V_T. Например, $\# \in V_T$. В формальных грамматиках множество V_T называют множеством ***терминальных символов***. Информационная лента исполняет функции ***внешней памяти*** машины Тьюринга.

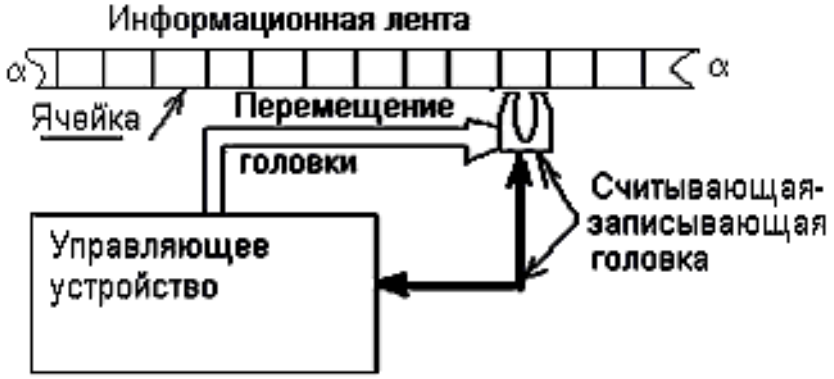


Рис.1. **Машина Тьюринга**

Считывающая-записывающая головка обозревает только одну ячейку информационной ленты, передает информацию о ее содержимом в управляющее устройство, и по указанию последнего сохраняет или изменяет содержимое ячейки.

Управляющее устройство представляет собой механизм, который на каждом шаге вычисления находится в одном из множества состояний $Q = \{q_1, q_2,..., q_m\}$. В зависимости от состояния q_i и считанного символа a_j управляющее устройство выдает команду на стирание или запись символа в обозреваемую ячейку, перевод управляющего устройства в новое состояние и перемещение головки на соседнюю ячейку информационной ленты. Поэтому состояния управляющего устройства называют «памятью машины Тьюринга», так как машина помнит все промежуточные состояния, которые привели машину из состояния q_0 в состояние q_i. С позиции формальных грамматик множество сим-

волов, описывающих состояния управляющего устройства, есть множество ***нетерминальных символов***. Среди всех состояний управляющего устройства следует выделить q_o— начальное состояние («старт») и q_k — конечное состояние («стоп»), что облегчит составление протоколов машин Тьюринга и композицию нескольких машин Тьюринга. Для описания перемещений головки относительно информационной ленты введем дополнительный алфавит D={П, Л, С}, где П — означает перемещение головки вправо на одну ячейку информационной ленты, Л — влево на одну ячейку и С — останов.

Работа машины Тьюринга состоит в многократном повторении следующего цикла элементарных действий:

действие первое: считывание символа a_j, находящегося под считывающей головкой;

действие второе*:* поиск команды, отвечающей текущему состоянию управляющего устройства q_i, и считанному символу a_j , т.е.

$q_j\ a_j => q_l\ a_m\ D$;

действие третье*:* исполнение найденной команды, т.е. запись в обозреваемую ячейку символа a_m, перевод управляющего устройства в состояние q_i и перемещение головки на соседнюю ячейку информационной ленты D.

Эти три действия представляют одну элементарную команду. Последовательность команд для реализации процесса вычисления представляют ***программу алгоритмического процесса*** или ***протокол машины Тьюринга***. Следует отметить, что никакие две команды не могут иметь одинаковую пару текущего состояния q_i и считываемого символа a_j, т.е. пары (q_i a_j). Машина Тьюринга останавливается только в том случае, если на очередном шаге управляющее устройство генерирует состояние q_k. Результатом работы машины Тьюринга будет заключительное слово на информационной ленте.

2.1. Описание машины Тьюринга

Математическая модель машины Тьюринга имеет вид:

$$T=<V_T, Q.\ D, \varphi, \psi, \xi >,$$

где $V_T=\{a_i, a_2,..., a_n\}$ - символы внешней памяти,

$Q=\{q_o, q_i, q_2...., q_m\}$ - символы внутренней памяти (q_0 -начальное , q_i – текущее, q_k –заключительное состояния),

$D = \{П.\ Л, С\}$ - символы перемещения считывающей - записывающей головки (П - вправо, Л - влево, С - стоп),

$\varphi: Q \otimes V_T => V_T$ - функция выхода,

$\psi: Q \otimes V_T => Q$ - функция переходов,

$\xi: Q \otimes V_T => D$ - функция перемещения головки.

Описание машины Тьюринга есть последовательность символов на информационной ленте, положение считывающей – записывающей головки относительно ячейки информационной ленты и текущее состояние управляющего устройства. Такое описание называют ***конфигурацией*** машины Тьюринга:

$$K = \alpha\ q_i\ \beta,$$

где α - слово (или последовательность символов), расположенное слева от считывающей – записывающей головки, β - слово, расположенное под и справа от считывающей - записывающей головки; q_i — текущее состояние машины Тьюринга. Символ ‘a’, находящийся в ячейке непосредственно под считывающей - записывающей головкой, является первым символом слова β. К не заключительной конфигурации может быть применима только одна команда, которая переводит машину в новую конфигурацию. Так реализуется дискретность и детерминизм алгоритмического процесса. Для удобства анализа вычислительных алгоритмов математик Пост предложил ограничить множество символов внешнего алфавита V_T двумя символами, т.е. $V_T = \{|, \#\}$, где "|" (палочка) есть символ унарного кода числа, а "#" (диеза или решетка) есть символ пробела между числами, представленными в унарном коде. При этом любое целое положительное число может быть записано на информационной ленте последовательностью палочек, как это представлено в табл. 1.

Для упорядочения протоколов информационную ленту ограничивают только в одну сторону, т. е. существуют левые и правые полуленты. В зависимости от используемой полуленты приняты различные схемы записи конфигураций машины Тьюринга (табл. 2).

Таблица 1

Число в десятичной системе счисления	"Слово" в унарном коде на информационной ленте
0	$\| = \|^{o}$
1	$\|\| = \|^{1+1}$
2	$\|\|\| = \|^{2+1}$
i	$\|\|\| \ldots \| = \|^{i+1}$

Таблица 2

Стандартная конфигурация	Информационная полулента	
	правая	левая
Начальная	$q_0\|^{x\ 1+1}\#\ \|^{x\ 2+\ 1}\#\ldots\#\|^{x\ n+\ 1}$	$\|^{x\ 1+1}\#\ \|^{x\ 2+\ 1}\#\ldots\#\|^{x\ n} q_n\|$
Конечная	$q_k\|^{y\ +\ 1}$	$\|^{y} q_k\|$

Работу машины Тьюринга удобно описывать протоколом, таблицей и/или графом. При протокольной записи все команды должны быть записаны упорядоченным списком*. На заключительном шаге должно быть получено значение заданной функции $y=f(x_1, x_2,..., x_n)$.

Например,

1) $q_o a_i \text{—>} q_i a_j D$,

2) $q_i a_k \text{—>} q_j a_i D$,

.....

i) $q_| a_m \text{—>} q_k a_n C$.

* см. [5], [7].

При табличном описании каждая строка имеет имя текущего и начального состояний машины, а столбец – имя символа внешней памяти. Тогда элементами таблицы являются правые части команд $q_j a_i D$ (табл.3).

Таблица 3

Состояние	Символы V_T			
	a_i	$a_к$	...	a_m
q_n	$q_i a_j D$	...	…	…
q_i	...	$q_j a_i D$	…	...
...	...	...	...	...
q_i	...	...	...	$q_k a_n C$

Табличная форма описания машины более компактна и позволяет применить матричные методы анализа для оптимизации структуры алгоритма. При описании машины Тьюринга графом вершинами являются состояния управляющего устройства, а дугами — переходы в те состояния, которые предусмотрены командой. При этом на дуге над символом «/» указывают ***считываемый символ***, а под символом «/» - ***записываемый символ*** на информационную ленту и команду на перемещение головки (рис. 2).

Рис. 2. Граф поведения машины Тьюринга

2.2. Примеры машин Тьюринга

Рассмотрим машины Тьюринга (м. Т.), исполняющие некоторые примитивно-рекурсивные функции.

T_1:= вычисление функции C_n=0 на правой полу ленте: $q_o|^{x+1}\#\rightarrow q_k|\#$.

Пусть $V_т=\{|, \#\}$.

Протокол, табл. 4 и рис. 3 отображают процесс вычисления функции $C_n = 0$.

Таблица 4

Протокол T_1:	$q \in Q$	Символы V_T	
		|	#
1) $q_o\|\rightarrow q_1\#$ П,	q_o	q_1 # П	—
2) $q_1\|\rightarrow q_1\#$ П,	q_1	q_1 # П	q_2 #Л
3) $q_1\#\rightarrow q_2\#$ Л,	q_2	—	q_k |C
4) $q_2\#\rightarrow q_k\|C$.			

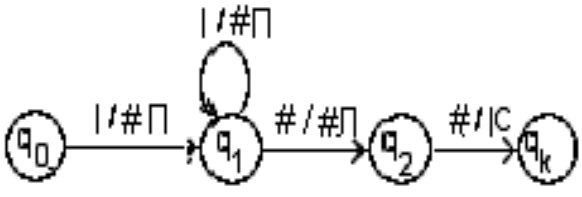


Рис. 3. Граф алгоритма м. Т. функции $C_n = 0$

Пусть x = 3. Тогда для $C_1(3)=0$ на информационной ленте машины Тьюринга будут проведены следующие преобразования:

x= 3 x= 0

#| | |# ⇒ ##| | |# ⇒ ###| |# ⇒ ####| # ⇒ ###### ⇒ ###### ⇒ #### | #

$q_n\uparrow\rightarrow$П $q_1\uparrow\rightarrow$П $q_1\uparrow\rightarrow$П $q_1\uparrow\rightarrow$П $q_1\uparrow\rightarrow$Л $q_2\uparrow$ П ← $q_k\uparrow$c

T_2:= вычисление функции $C_n=1$ на правой полуленте: $q_o|^{x+1}\#\rightarrow q_k||\#$.
Протокол T_2, табл. 5 и рис. 4 отображают процесс вычисления функции $C_n=1$.

Таблица 5

Протокол T_2:
1) $q_o|\rightarrow q_1\#$ П,
2) $q_1|\rightarrow q_1\#$ П,
3) $q_1\#\rightarrow q_2\#$ |Л,
4) $q_2\#\rightarrow q_k|$C.

q∈Q	Символы V_T	
	|	#
q_o	q_1 #П	—
q_1	q_1 #П	q_2 |Л
q_2	—	q_1 |C

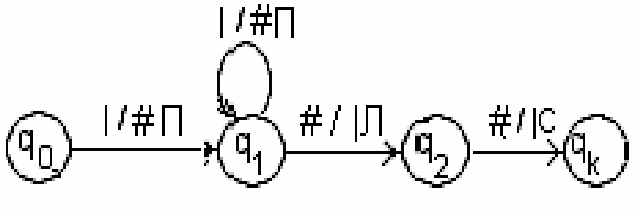


Рис. 4. Граф алгоритма м. Т. функции $C_n = 1$

T_3:= вычисление функции $\lambda(x)=x+1$ на правой ленте: $q_o|^{x+1}\# \rightarrow q_k|^{(x+1)+1}\#$.
Протокол T_3, табл. 6 и рис. 5 отображают процесс вычисления $\lambda(x)=x+1$.

Протокол T_3:
1) $q_o| \rightarrow q_1|$П,
2) $q_1| \rightarrow q_1|$П,
3) $q_1\# \rightarrow q_2|$Л,
4) $q_2| \rightarrow q_2|$Л,
5) $q_2\# \rightarrow q_3\#$ П,
6) $q_3| \rightarrow q_k|$C.

Таблица 6

q∈Q	Символы V_T	
	|	#
q_i	q_1 | П	-
q_1	q_1 | П	q_2 | Л
q_2	q_2 | Л	q_3 # П
q_3	q_k | С	-

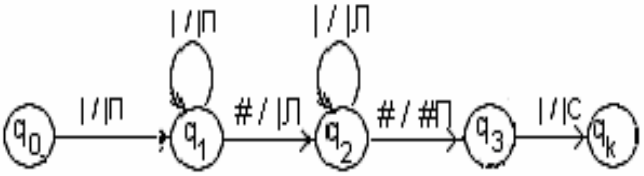


Рис. 5. Граф алгоритма м. Т. функции $\lambda(x)=x+1$

Пусть x=3. Тогда для $\lambda(3)=3+1=4$ на информационной ленте будут проведены следующие преобразования машиной Тьюринга:

→x=3 x=4

#| | |# ⇒ #| | |# ⇒#| | |# ⇒#| | |# ⇒# | | | |#⇒#| | | |#⇒# | | | |#

$q_0\uparrow$П→ $q_1\uparrow$П→ $q_1\uparrow$П→ $q_1\uparrow$П→ $q_1\uparrow$Л ← $q_2\uparrow$Л ← $q_2\uparrow$Л

⇒# | | | |#⇒# | | | |#⇒# | | | |#⇒# | | | |#⇒# | | | |#.

← $q_2\uparrow$Л ← $q_2\uparrow$Л $q_2\uparrow\Pi\rightarrow$ $q_3\uparrow$ $q_k\uparrow$c

T_4:= вычисление функции I_n^m на правой ленте:

$$q_0|_1^{x1+1}\#|_2^{x2+1}\#...\#|_m^{xm+1}\#...\#|_n^{xn+1}\# \rightarrow q_k|_m^{xm+1}\#,$$

где $|_i$ - слово на i-м месте информационной ленты, $|^{xi+1}$ – число, равное x_i.

Чтобы найти i-е слово, следует установить счетчик $|_{m-1}$, указывающий начало слова $|_m^{xm+1}$ в $q_0|_{m-1}\ \#|_1^{x1+1}\#|_2^{x2+1}\#\ ...\#|_m^{xm+1}\#|_n^{xn+1}$. Это позволяет при каждом проходе вправо удалять один символ в слове $|_{m-1}$ и удалять одно слово $|_{i-1}$, предшествующее слову $|_m^{xi+1}$. Поведение машины описано в табл. 7 и графом на рис. 6.

Таблица 7

q∈Q	Символы V_T		
	|	#	)
q_o	q_1 #П	-	-
q_1	q_1 |П	q_2)П	-
q_2	q_2 #П	q_3)Л	-
q_3	-	q_3 #Л	q_4)Л
q_4	q_4 |Л	q_5 # П	-
q_5	q_6 #П	-	q_8 #П
q_6	q_6 |П	-	q_7)П
q_7	-	q_7 #П	q_2 #П
q_8	-	q_8 #П	q_9 #П
q_9	q_9 |П	q_{10} #П	-
q_{10}	q_{10} #П	q_{11} #П	-
q_{11}	q_{10} #П	q_{12} #Л	-
q_{12}	q_{13} |Л	q_{12} #Л	-
q_{13}	q_{13} |Л	q_{14} #П	-
q_{14}	q_k |С		-

Пусть $V_т = \{|, \#,)\}$.

Считывающая головка находится под первым символом счетчика $|_{m-1}$. На рис. 6 приведен граф, реализующий базовую функцию $I_{m,n}$. При подаче команды «старт» стирается первый символ слова $|_m$ и начинается перемещение головки вправо. После прочтения всех символов $|_{m-1}$ по команде $q_1\#\rightarrow q_2)П$ ставится скобка, закрывающая оставшиеся символы $|_{m-1}$, и головка перемещается на слово $|_1^{x1+1}$. По команде $q_2|\rightarrow q_2\#П$ стираются все символы слова $|_1^{x1+1}$ и ставится закрывающая скобка $q_2\#\rightarrow q_3)Л$. Начинается перемещение считывающей головки влево за очередным символом слова $|_{m-2}$. Так организован цикл.

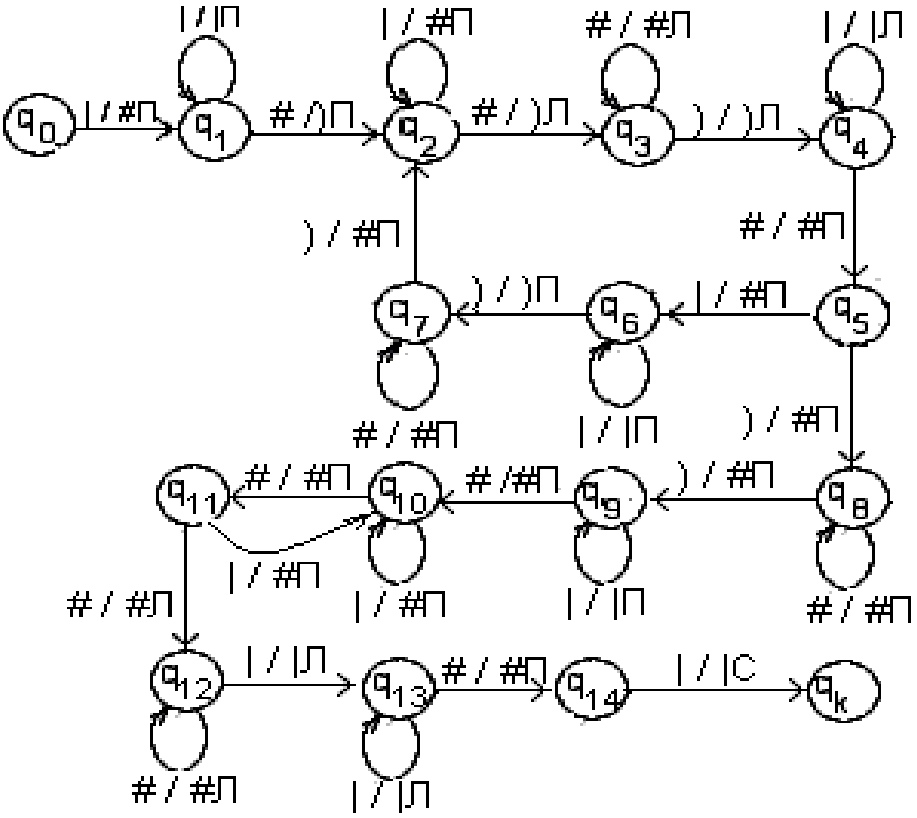


Рис. 6. Граф алгоритма м. Т. функции I_n^m

Затем управляющее устройство стирает второе слово, третье и т. д. пока есть символы "|" в счетчике. После стирания всех символов в счетчике по команде $q_5)\rightarrow q_8\#П$ головка стирает закрывающую скобку, пробегает все пробелы "#", по команде $q_8)\rightarrow q_9\#П$ фиксирует место слова $|_m^{xm+1}$, пробегает все символы слова $|_m^{xm+1}$ и продолжает удалять все слова справа от слова $|_m^{xm+1}$по командам $q_{10}|\rightarrow q_{10}\#П$ и $q_{11}|\rightarrow q_{10}\#П$. Если при чтении ленты в соседних ячейках будут символы "#", то конец и головка возвращается на первый символ слова $|_m^{xm+1}$ по командам: $q_{11}\#\rightarrow q_{12}\#Л$, $q_{12}\#\rightarrow q_{12}\#Л$, $q_{12}|\rightarrow q_{13}|Л$, $q_{13}|\rightarrow q_{13}|Л$, $q_{13}\#\rightarrow q_{14}\#П$ и $q_{14}|\rightarrow q_k|С$.

T_5:= вычисление примитивно рекурсивной функции предшествования $\lambda^{-1}(x) = x\dot{-}1$ на правой ленте: $q_i\,|^{x+1}\#\ \rightarrow\ q_k\,|^{(x+1)-1}\#$.

Протокол, табл.8 и рис. 7 отображают процесс вычисления $\lambda^{-1}(x) = x\dot{-}1$.

Пусть $V_т=\{|, \#\}$.

Таблица 8

Протокол T_5:	$q\in Q$	Символы V_T	
		|	#
1) $q_o\|\rightarrow q_1\|П$,	q_o	q_1 | П	-
2) $q_1\|\rightarrow q_1\|П$,	q_1	q_1 | П	q_2 # Л
3) $q_1\#\rightarrow q_2\#Л$,	q_2	q_3 # Л	-
4) $q_2\|\rightarrow q_3\#Л$,	q_3	q_3 | Л	q_4 # П
5) $q_3\|\rightarrow q_3\|Л$,	q_4	q_k | С	-
6) $q_3\#\rightarrow q_4\#П$,			
7) $q_4\|\rightarrow q_k\|С$.			

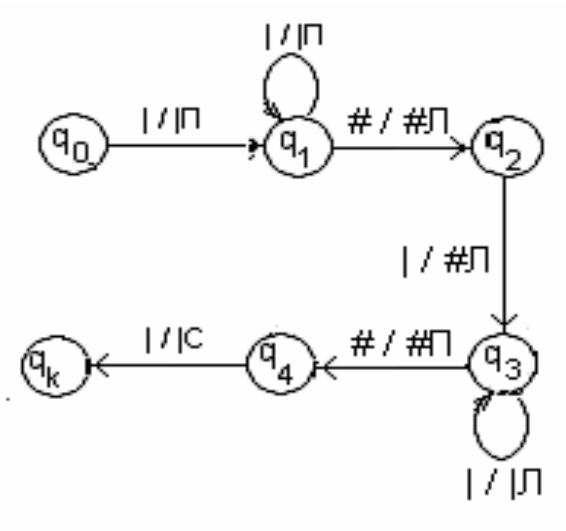


Рис. 7. Граф алгоритма м. Т. функции $\lambda^{-1}(x) = x\dot{-}1$

T_6:=копирование m раз слово $|^{x+1}$на правой ленте:

$q_0|^{x+1}\#\ \rightarrow q_k\,|_1^{x+1}\,\#|_2^{x+1}\#|_3^{x+1}\#\ ...\#|_m^{x+1}\#$.

Для того чтобы выполнить m копий, необходимо установить на информационной ленте счетчик числа копий $|_m$, т.е. изменить начальную конфигурацию машины так: $q_0|_m\#|^{x+1}\#\ \rightarrow q_k|_1^{x+1}\#_2^{x+1}\#|_3^{x+1}\ \#\ ...\#|_m^{x+1}\#$.

При каждом проходе вправо в слове $|_m$ удаляется один символ, затем выполняется обычное копирование слова $|^{x+1}$. Пометка переносимого символа слова $|^{x+1}$ выполняется командой $q_2|\rightarrow q_3(П$, а закрытие слова $|^{x+1}$ - командой $q_3\#\rightarrow q_4)П$. После переноса всех символов слова $|^{x+1}$ головка возвращается на самый левый символ слова $|_m$ и стирает его. Так организован цикл. Копирование

следующих m копий будет продолжаться до тех пор, пока не будут удалены все символы в слове 1_m. После этого следует удалить символы (,) и поставить головку машины в начало первого слова $|^{x+1}$. Множество команд, описывающих поведение машины Тьюринга, представлено табл. 9 и графом на рис. 8.

Пусть $V_т$.={|, #, (,)}.

Таблица 9

q∈Q	Символы V_T			
	|	#	(	)
q_o	q_1#П	-	-	q_8 #П
q_1	q_1 |П	q_2)П	-	q_2)П
q_2	q_3 (П	-	q_2)П	q_2)П
q_3	q_3 |П	q_4)П	-	q_4)П
q_4	q_4 |П	q_5 |Л	-	-
q_5	q_5 |Л	-	q_6 (П	q_5)Л
q_6	q_3 (П	-	-	q_7)Л
q_7	q_7 |Л	q_0#П	q_7 (Л	q_7)Л
q_8	q_9 |Л	-	q_8 (П	q_8)П
q_9	q_k |С	q_9#П	q_9 |Л	q_9#Л

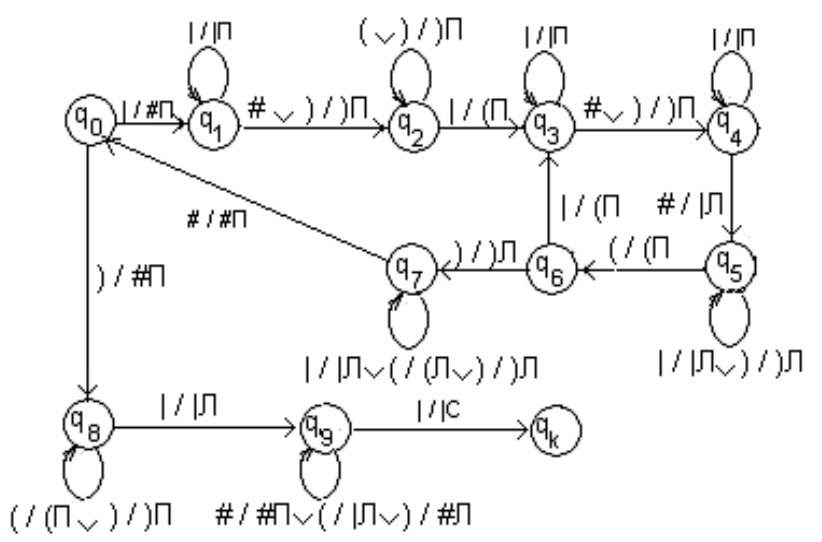


Рис. 8. Граф алгоритма м. Т. копирования слова $|^{x+1}$ m раз

T_7:= сравнение длины двух слов на правой полуленте:

$T_7^{(1)}$: $q_o 1^{x+1} \#|^{y+1} \rightarrow q_{k1}|^{(x-y)}$ при x > y;

$T_7^{(2)}$: $q_o 1^{x+1} \#|^{y+1} \rightarrow q_{k3}|$ при x = y;

$T_7^{(3)}$: $q_o |^{x+1} \#|^{y+1} \rightarrow q_{k2}|^{(y-x)}$ при x < y.

Таблица 10

q∈Q	Символы V_T		
	|	#	)
q_o	q_1 #П	-	-
q_1	q_1 |П	q_2)П	q_2)П
q_2	q_2 |П	q_3 #Л	-
q_3	q_4 #Л	-	q_6 #Л
q_4	q_4 |Л	q_5 #П	q_4)Л
q_5	q_1 #П	-	q_8 #П
q_6	q_6 |Л	q_7 #П	-
q_7	q_{k1} |С	-	-
q_8	q_{k2} |С	q_{k3} |С	-

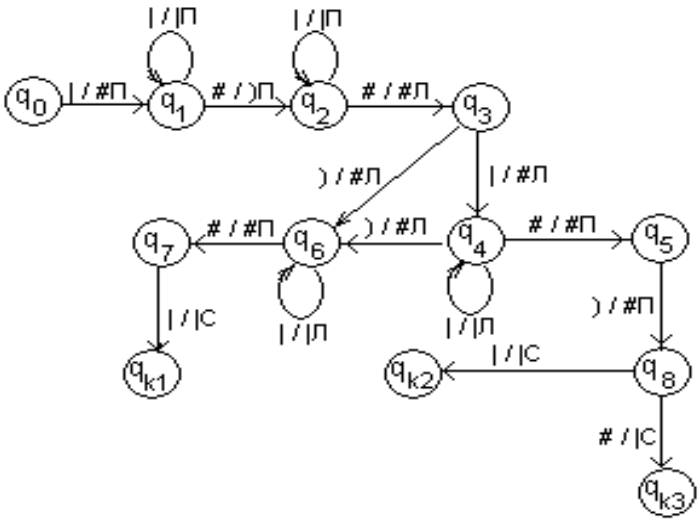


Рис. 9. Граф алгоритма м. Т. сравнения длины двух слов

Особенность задачи состоит в том, что слово (x+1) может быть больше, меньше или равным слову (y+1). При этом машина Тьюринга должна переходить в три различных состояния для организации ветвящегося процесса вычисления. Пусть Vт={|, #,)}. Поведение машины Тьюринга описано табл.10 и графом на рис. 9.

T_8:= перестановка слов: $q_o|^{x+1}\#\,|^{y+1}\# \rightarrow q_k|^{y+1}\#\,|^{x+1}\#$.
Протокол, табл. 11 и рис. 10 отображают процесс перестановки слов.

Пусть Vт={|, #, (,)}.

Протокол T_8:

1)$q_o| \rightarrow q_1$(П, 12)$q_4)\rightarrow q_4)$Л,
2)$q_1| \rightarrow q_1|$П, 13)$q_5|\rightarrow q_1$(П,
3)$q_1\#\rightarrow q_2)$П; 14)$q_5)\rightarrow q_6)$П,
4)$q_1) \rightarrow q_2)$П, 15)$q_6|\rightarrow q_6|$П,
5)$q_2| \rightarrow q_2|$П, 16)$q_6)\rightarrow q_7\#$ Л,
6)$q_2\# \rightarrow q_3)$П, 17)$q_7|\rightarrow q_7|$Л,
7)$q_2)\rightarrow q_3)$П; 18)$q_7(\rightarrow q_7\#$Л,
8)$q_3|\rightarrow q_3|$П, 19)$q_7)\rightarrow q_7\#$Л,
9)$q_3\#\rightarrow q_4|$Л, 20)$q_7\#\rightarrow q_8\#$П,
10)$q_4|\rightarrow q_4|$Л, 21)$q_8\#\rightarrow q_8\#$П,
11)$q_4(\rightarrow q_5$(П, 22)$q_8|\rightarrow q_k|$C.

Таблица 11

q∈Q	Символы V_T			
	|	#	(	)
q_o	q_1 (П	-	-	-
q_1	q_1 |П	q_2)П	-	q_2)П
q_2	q_2 |П	q_3)П	-	q_3)П
q_3	q_3 |П	q_4 |Л	-	-
q_4	q_4 |Л	-	q_5 (П	q_4)Л
q_5	q_1 (П	-	-	q_6)П
q_6	q_6 |П	-	-	q_7 #Л
q_7	q_7 |Л	q_8 #П	q_7 #Л	q_7 #Л
q_8	q_k |C	q_8 #П	-	-

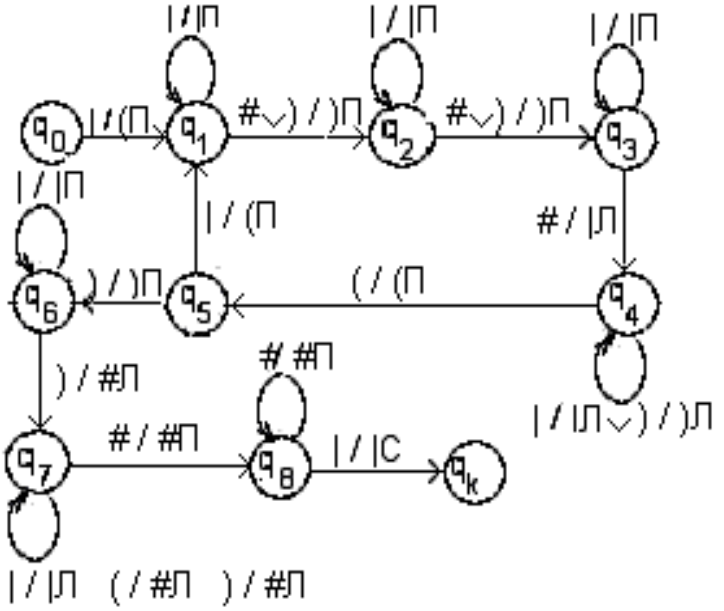


Рис. 10. Граф алгоритма м. Т. перестановки слов.

T_9:= устранение пробелов между словами $q_o|^{x1+1}\#\#...\#|^{x2+1}\#\rightarrow q_k|^{x1+1}\#\,|^{x2+1}\#$.
Пусть Vт={|, #,)}.

Таблица 12

q∈Q	Символы V_t		
	|	#	)
q_o	q_1 |П	-	-
q_1	q_1 |П	q_2)П	-
q_2	q_3 |П	q_2 #П	q_7 #Л
q_3	q_3 |П	q_4 #Л	q_4 #Л
q_4	q_5)Л	-	-
q_5	q_5 |Л	q_5 #Л	q_6)П
q_6	q_6 |П	q_2 |П	-
q_7	q_7 |Л	q_7 #Л	q_8 #Л
q_8	q_8 |Л	q_9#П	-
q_9	q_k|С	-	-

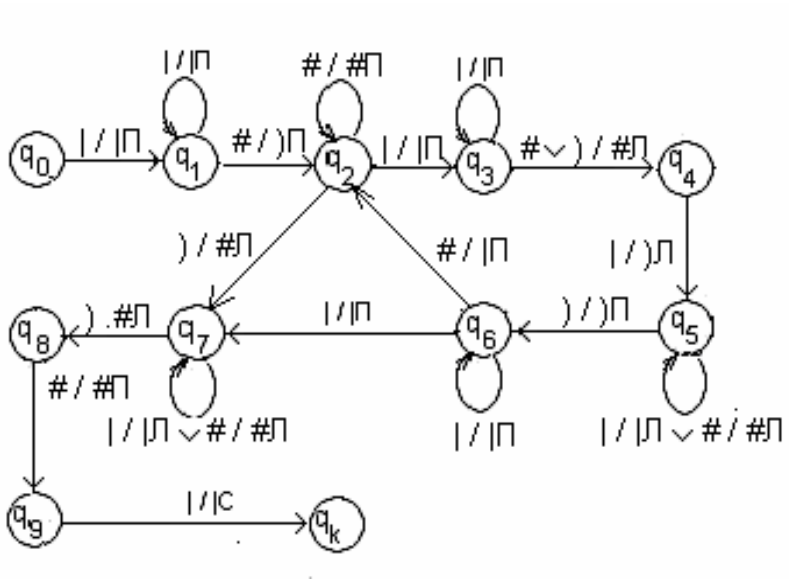


Рис. 11. Граф алгоритма м. Т. устранения пробелов между словами

Табл. 12 и рис. 11 отображают процесс устранения пробелов между словами. После просмотра первого слова символ «#» замещается «)» для ограничения переноса влево символов второго слова. Последний символ второго слова также замещается «)» для ограничения движения считывающей головки вправо. После переноса всех символов второго слова устраняются «)», и считывающая головка возвращается на первый символ первого слова.

T_{10}:= поиск последнего слова на правой ленте:

$q_o|^{x1+1}\#|^{x2+1}\#... \# |^{xm+1} \rightarrow |^{x1+1}\#|^{x2+1}\#... \# q_k|^{xm+1}$.

Поиск последнего слова на правой ленте завершается после нахождения не менее двух ## (пробелов), что свидетельствует об окончании последовательности слов. Затем считывающая головка возвращается на первый символ последнего слова. На рис. 12 представлен граф, а в табл. 13 – протокол алгоритма.

Таблица 13

q∈ Q	Символы V_t	
	|	#
q_o	q_1|П	-
q_1	q_1|П	q_2#П
q_2	q_1|П	q_3#Л
q_3	q_4|Л	q_3#Л
q_4	q_4|Л	q_5#П
q_5	q_k|С	-

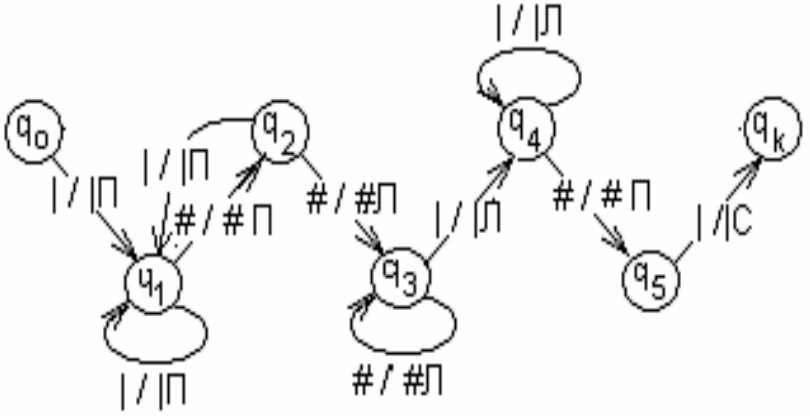


Рис. 12. Граф алгоритма м. Т. поиска последнего слова на правой ленте

T_{11}:= вычисление функции $f(x, y) = x \dot{-} y$:

$$q_0 |^{x+1} \# |^{y+1} \to \begin{cases} q_{k1} |^{(x-y)+1} \#, \text{ если } x > y, \\ q_{k2} |\#, \text{ если } x \le y. \end{cases}$$

Таблица 14

q∈Q	Символы V_t		
	|	#	)
q_o	q_1#П	-	-
q_1	q_1|П	q_2)П	q_2)П
q_2	q_2|П	q_3 #Л	-
q_3	q_4#Л	-	q_6 #Л
q_4	q_4|Л	q_5 #П	q_4)Л
q_5	q_1#П	-	q_8 #П
q_6	q_6|Л	q_7 #П	-
q_7	q_{k1}|C	-	-
q_8	q_8#П	q_9#П	-
q_9	-	q_{k2}|C	-

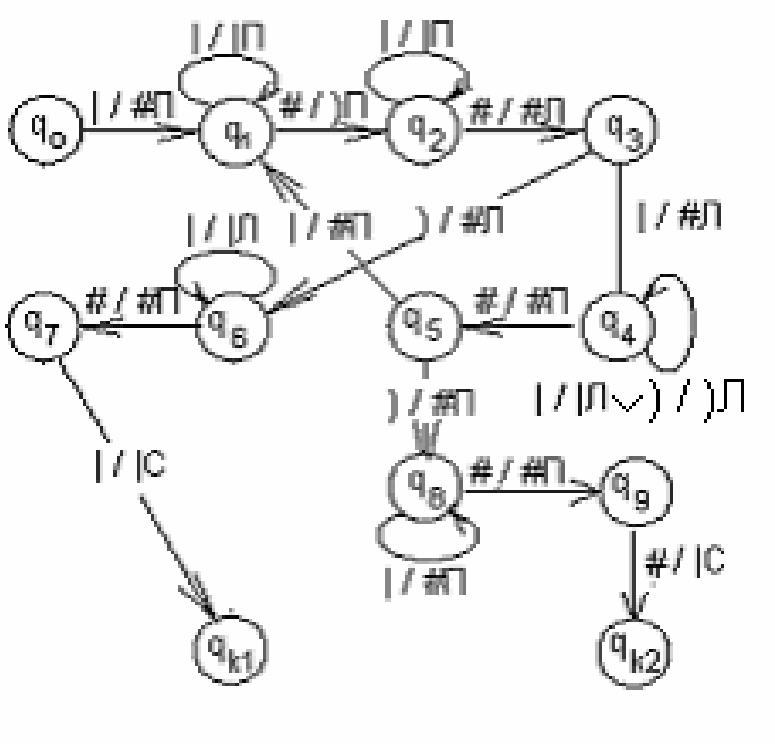


Рис. 13. Граф алгоритма м. Т. операции усеченного вычитания

Для вычисления усеченного вычитания можно использовать машины T_8 и T_1, изменив команды машины T_8: $q_8| \to q_{k2}|C$ и $q_8\# \to q_{k3}|C$ на команды машины T_1:$q_8| \to q_8\#П$, $q_8\#П \to q_8\#П$ (состояния q_1 и q_2 машины T_1 есть, соответственно, q_8 и q_9 машины T_{11}). В табл. 14 и на рис. 13 показан протокол решения усеченного вычитания.

T_{12}:= вычисление предиката P(x, y):

$$q_0 |^{x_1+1} \# |^{x_2+1} \# \to \begin{cases} q_{r1} ||\#, \text{ если } P(x_1, x_2) := \text{"истина"}, \\ q_{r2} |\#, \text{ если } P(x_1, x_2) := \text{"ложь"}. \end{cases}$$

Для этого можно использовать машины Тьюринга T_8, T_1 и T_2.
В табл. 15 и на рис. 14 даны протокол и граф алгоритма.

Таблица 15

$q \in Q$	Символы V_t		
	|	#	)
q_0	q_1#П	-	-
q_1	q_1|П	q_2)П	q_2)П
q_2	q_2|П	q_3#Л	-
q_3	q_4#Л	-	q_6#Л
q_4	q_4|Л	q_5)#П	q_4)Л
q_5	q_1#П	-	q_8#П
q_6	q_6|Л	q_7#П	-
q_7	q_7#П	q_{10}|Л	-
q_8	q_8#П	q_9#П	-
q_9	-	q_{k2}|С	-
q_{10}	-	q_{k1}|С	-

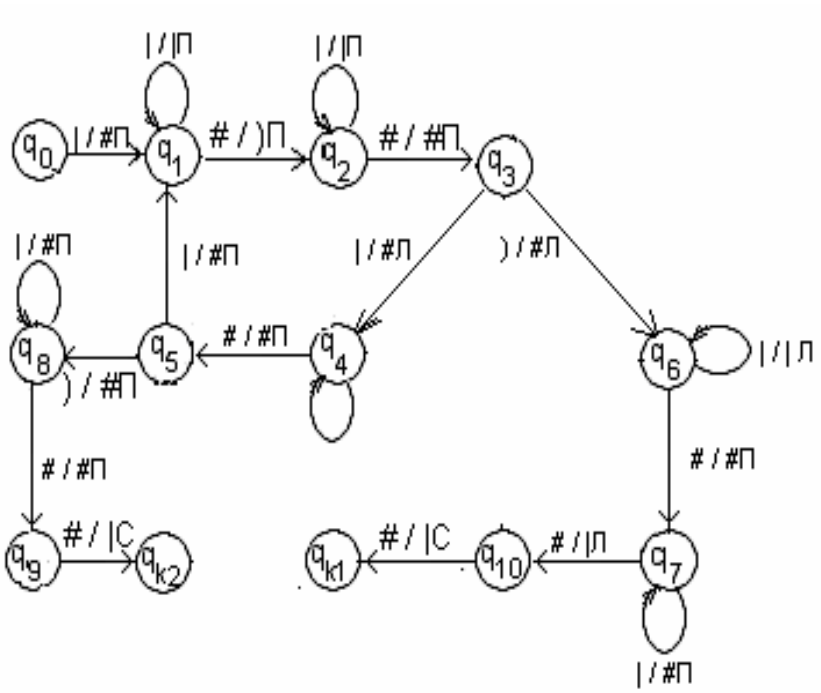


Рис. 14. Граф алгоритма м. Т. вычисления предиката

2.3. Композиция машин Тьюринга

Для графического изображения основных операций на схеме алгоритма приняты обозначения основных блоков, показанные на рис. 15.

Последовательное и/или параллельное соединение различных блоков позволяет организовать вычисление любых частично-рекурсивных функций. Общую схему соединения нескольких блоков от “начало” до “конец” называют ***блок-схемой алгоритма***.

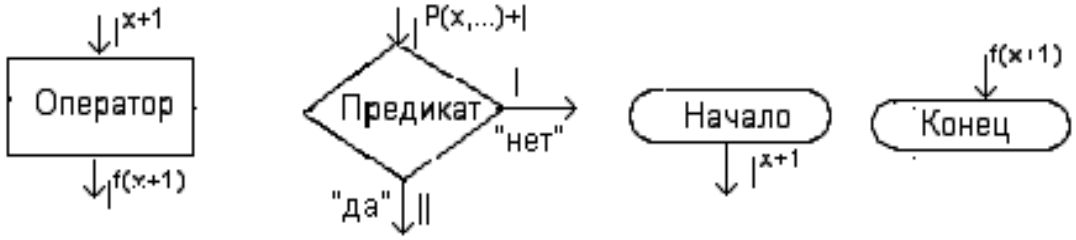


Рис. 15. Основные блоки схемы алгоритма

Оператор суперпозиции $S_n^m(h^{(m)}, g_m^{(n)})$. Если даны m-местная функция $h=(z_1, z_2,..., z_m, y)$ и m n-местных функций $g_i=(x_1, x_2,..., x_n, z_i)$, где $1 \le i \le m$, которые вычислимы по Тьюрингу, то функция $f=(x_1, x_2,..., x_n, y)$ также вычислима.

Алгоритм вычисления оператора суперпозиции:

шаг 1: вычислить по заданным значениям $(x_1, x_2,..., x_n)$ значения функций $g_i (x_1, x_2,..., x_n)$ для $1 \le i \le m$:

T_g: $q_0|^{x_1+1}\#|^{x_2+1}\#...\#|^{x_n+1} \rightarrow q_k|^{g_1(x_1+1,x_2+1,...,x_n+1)+1}\#|^{g_2(x_1+1,x_2+1,...,x_n+1)+1}\#...\#|^{g_m(x_1+1,x_2+1,...,x_n+1)+1}\#$;

шаг 2: вычислить для полученных значений g_i (x_1, x_2,..., x_n) значение функции $h(g_1, g_2,..., g_m)$:

T_h: $q_0|^{g_1(x_1+1,x_2+1,...,x_n+1)+1}\#|^{g_2(x_1+1,x_2+1,...,x_n+1)+1}\#...\#|^{g_m(x_1+1,x_2+1,...,x_n+1)+1}\# \rightarrow q_k|^{h(g_1,g_2...,g_m)+1}\#$;

шаг 3: выдать результаты расчетов:

$q_0|^{h(g_1, g_2,...,g_m)+1}\# \rightarrow q_k|^{f(x+1)+1}\#$.

На рис. 16 изображена композиция машин Тьюринга, реализующая оператор суперпозиции для n=1 и m=1.

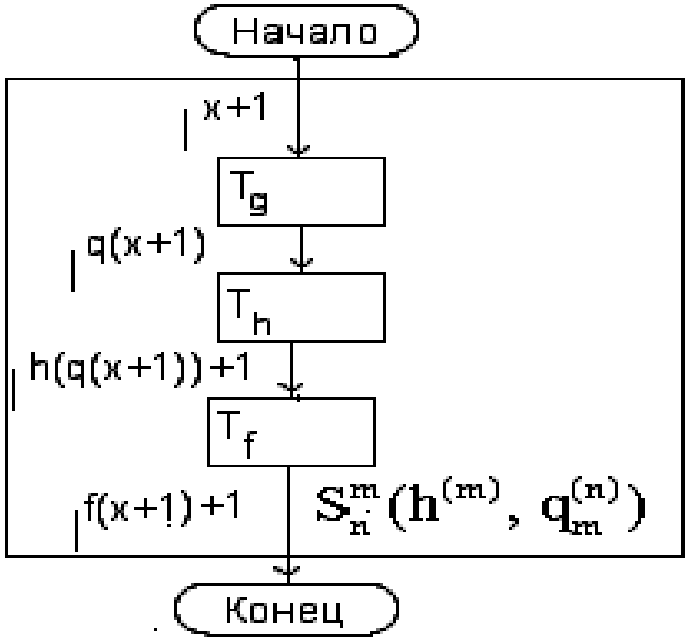


Рис. 16. Схема алгоритма оператора суперпозиции

Оператор рекурсии. Если даны n-местная функция $g(x_1, x_2,..., x_n)$ и (n+2)-местная функция $h(x_1, x_2,..., x_n, y, f(x_1, x_2,..., x_n, y))$, которые вычислимы на машине Тьюринга, то функция $f(x_1, x_2,..., x_n, y+1)=h(x_1, x_2,..., x_n, y, f(x_1, x_2,..., x_n, y))$ также вычислима на машине Тьюринга.

Алгоритм вычисления оператора рекурсии:

шаг 1: вычислить для заданного значения x функцию g(x);

T_g: $q_o|^{x+1}\#|^{y+1} \rightarrow q_k|^{x+1}\#|^{y+1}\#|^{g(x+1)+1}$, (если g является базовой функцией, то процесс ее вычисления приведен на машинах T_1, T_2, T_3);

шаг 2: ввести переменную i для счета символов слова (y+1):

T_i: $q_o|^{x+1}\#|^{y+1}\#|^{g(x+1)+1} \rightarrow q_k|^{i+1}\#|^{x+1}\#|^{y+1}\#|^{g(x+1)+1}$; установить i = 0;

шаг 3: вычислить значение функции h(x, i, f (x, i)):

T_h: $q_o|^{i+1}\#|^{x+1}\#|^{y+1}\#|^{g(x+1)+1} \rightarrow q_k|^{i+1}\#|^{x+1}\#|^{y+1}\#|^{h((x+1),\ i,\ f(x+1,\ i))+1}$;

шаг 4: проверить i = y;

а) если $i \neq y$, то принять i = i+1 и перейти к шагу 3;

b) если i = y перейти к шагу 5; (для проверки условия использовать T_{12});

шаг 5: удалить с ленты слова $|^{y+1}$, $|^{x+1}$, $|^{y+1}$:

$T_k : q_i|^{y+1}\#|^{x+1}\#|^{y+1}\#|^{h\,(x+1,\,y,\,f\,(x+1,\,y))+1}\# \rightarrow q_k|^{h\,(x+1),\;y,\;f\,(x\,+1,\;y))+1}\#$;

шаг 6: выдать результаты расчетов:

$q_0|^{h\,(x+1),\,y,\,f\,(x\,+1,\,y))+1}\# \;\rightarrow\; q_k|^{f\,(x\,+1,\;\,y+1)+1}\#$.

При реализации оператора рекурсии необходимо вычислять значения f(x, i+1) до тех пор, пока i≠y. При достижении i=y процесс вычисления прекращается и результат вычисления есть f(x, y+1)=h(x, y, f(x, y)). Если каждый шаг вычислительного процесса представить отдельной машиной Тьюринга, то их последовательное соединение (кроме машины, реализующей шаг 4) обеспечивает поиск значения функции f(x, y). Результат исполнения шага 4 должен поступить на входы машин T_i или T_k, т.е. машина имеет два выхода. На рис. 17 приведена схема соединения машин Тьюринга для оператора примитивной рекурсии.

Если функции g(x) и h(x, y, f(x, y)) не базовые, но вычислимые по схеме примитивной рекурсии, то функция f(x, y) также вычислима на машине Тьюринга. Вычислимые функции, значения которых определены не для всех значений независимых переменных аргумента, называют частичными. Следовательно, вычислимые функции, для которых область определения ограничена, называют ***частично вычислимыми функциями***. Если частично вычислимая функция определена на множестве целых положительных чисел и принимает значение на том же множестве, то ее называют частично рекурсивной функцией. Вычислимую функцию, всюду определенную на множестве целых положительных чисел и принимающую значение на том же множестве, называют общерекурсивной функцией. Для иллюстрации этого алгоритма рассмотрим случай для n=1, т.е. g(x), h(x, y, f(x, y)) и f(x, y+1)= h(x, y, f(x, y)) (рис. 17).

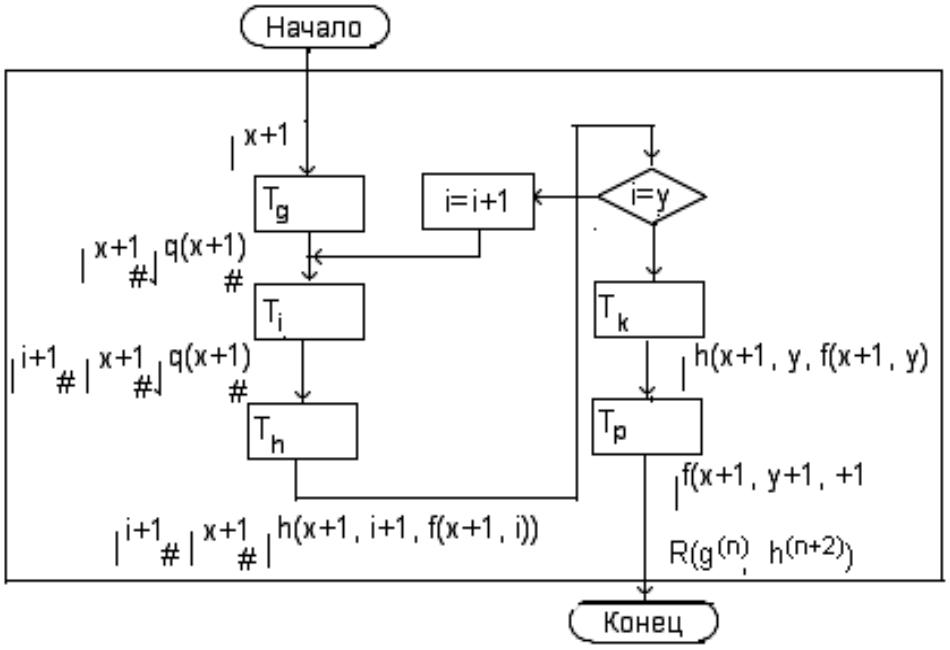


Рис. 17. Схема алгоритма оператора рекурсии

Оператор минимизации. Для поиска наименьшего корня функции $\varphi(x_1, x_2,..., x_n)=\mu_y(f(x_1, x_2,..., x_n, y)=0)$ необходимо организовать последовательное приращение вспомогательного аргумента y=0, 1, 2.... и сравнивать значение функции $f(x_1, x_2,..., x_n, y)$ с базовой функцией $C_n=0$.

Алгоритм вычисления оператора минимизации (см. рис. 18):

шаг 1: ввести переменную i в функцию $f(x_1, x_2,\ldots, x_n, y)$:

T_i: $q_o| ^{f(x1+1\#\ x2+1\#...\ Xn+1,\ y+1)} \rightarrow q_k| ^{f(x+11\#\ x2+1\#...\ Xn+1\#\ i+1)}\#$; установить i = 0;

шаг 2: вычислить значение функции $f(x_1, x_2,..., x_n, i)$:

T_f: $q_o| ^{f(x+11\#\ x2+1\#....\ Xn+1\#\ i+1)}\# \rightarrow q_k| ^{f(x+11\#\ x2+1\#...\ Xn+1\#\ i+1)}\#$;

- если $f(x_1, x_2,\ldots, x_n, i)\neq 0$, то принять i=i+1 и перейти к шагу 2;
- если $f(x_1, x_2,\ldots, x_n, i)=0$, то конец;

шаг 6: выдать результаты расчетов:

$|^{i+1}= |^{y+1}= |^{\varphi(x+11\#\ x2+1\#...\ Xn+1)}\#$.

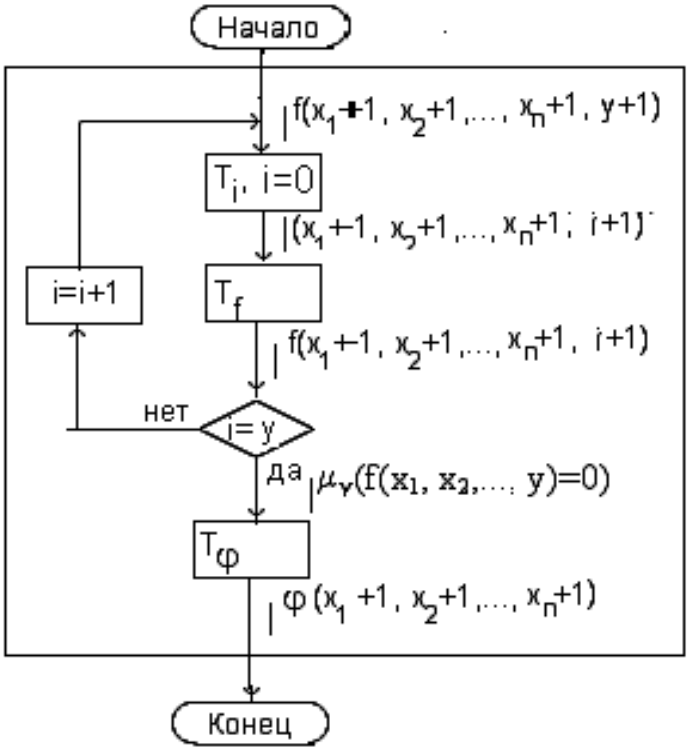


Рис. 18. Схема алгоритма оператора минимизации

Оператор итерации. Условием итерации является число повторений n. Схема итерации (рис. 19): $f(i) = \begin{cases} f(0) = 0, \\ f(i+1) = h(f(i)). \end{cases}$

Алгоритм вычисления оператора итерации:

шаг 1: ввести необходимое число шагов n и функцию h(f(i));

шаг 2: вычислить значение функции h(f(i)) для $1\leq i \leq n$:

- если $i\neq n$, то перейти к шагу 2;
- если i=n, то конец;

шаг 3: выдать результаты расчетов:

$J^n(h)=|^{f(n)+1}$.

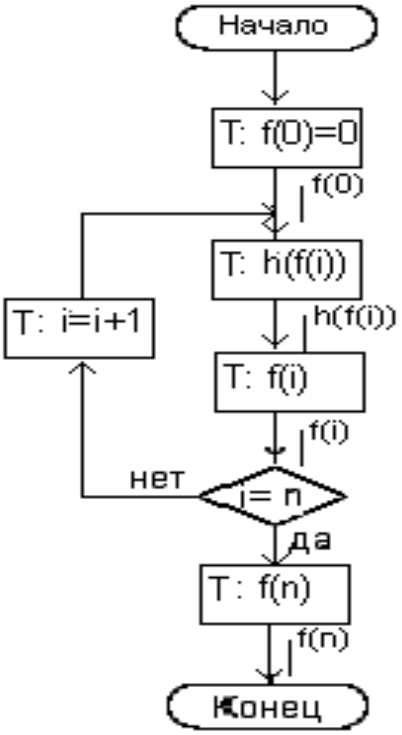


Рис. 19 Схема алгоритма итерации

В приведенных примерах показано, что базовые функции и элементарные операторы вычислимы на машине Тьюринга, т.е. любая частично рекурсивная функция может быть вычислена на машине Тьюринга и, наоборот, любая вычислимая на машине Тьюринга функция является частично рекурсивной.

2.4. Нумерация алгоритмов

Для успешного использования левой и правой полулент машины Тьюринга была предложена система кодирования символов трех алфавитов.

Таблица 16

Символ	Знак	Код	Число нулей
Сдвига головки	вправо - П	$10=10^1$	1
	влево - Л	$100=10^2$	2
	останов - С	$1000=10^3$	3
Внешней памяти	#	$10000=10^4$	4
	a_1 или |	$1000000=10^6$	6
	a_2	$100000000=10^8$	8
	…	…	…
	a_i	$100\ldots00=10^{(2\cdot i+4)}$	$(2\cdot i+4)$
Внутренней памяти	q_0	$100000=10^5$	5
	q_k	$10000000=10^7$	7
	q_1	$1000000000=10^9$	9
	q_2	$100000000000=10^{11}$	11
	…	…	…
	q_j	$100\ldots00=10^{(2\cdot i+5)}$	$(2\cdot i+5)$

Согласно этой системе (табл. 16) символы «сдвига головки» были представлены кодами 10^1, 10^2, 10^3, символы «внешней памяти» - кодами с числом нулей равном $(2\cdot i + 4)$, символы «внутренней памяти» – кодами с числом нулей $(2\cdot i + 5)$.Символ # интерпретируются как пробел между числами или словами. Тогда любой команде (i) машины Тьюринга, имеющей вид $q_m a_s \rightarrow q_l a_u D$ может быть сопоставлен код: код (i)= код(q_m)код(a_s)код(q_l)код(a_n)код(D), в котором коды всех символов приписаны друг к другу без каких-либо пробелов. Разделителем слов или чисел являются 1, так как код любого символа начинается с 1, а их различие определяется количеством нулей в коде.

На левой полуленте записывают протокол машины Тьюринга, а на правой – ее текущую конфигурацию. В процессе вычисления заданной функции на правой полуленте определяется левая часть команды $q_m a_s$, где q_m текущее состояние управляющего устройства, a_s – символ на информационной ленте под считывающей головкой. Затем на левой полуленте находится соответствующая команда $q_m a_s \rightarrow q_l a_u D$. После этого машина Тьюринга записывает в обозреваемую ячейку символ a_u, переводит управляющее устройство в состояние q_l и перемещает головку не более как на одну ячейку вправо или влево.

Например, для машины T_1 протокол будет записан на левой полу ленте так: код(T_1)=$10^5 10^6 10^9 10^4 10^1 10^9 10^6 10^9 10^4 10^1 10^9 10^4 10^{11} 10^4 10^2 10^{11} 10^4 10^7 10^6 10^3$.

Указанное кодирование является алгоритмической процедурой. Каждые пять групп единиц и нулей представляют одну команду. Следовательно, любой функции, вычисляемой на машине Тьюринга, может быть сопоставлен код (T_i) и по коду (T_i) можно выделить алгоритм вычисляемой функции. Так может быть организована нумерация алгоритмов машины Тьюринга.

В 70-е годы XX века была предложена алгоритмическая модель, представляющая собой идеализированную ЭВМ. Эта модель состоит из бесконечного числа регистров R1, R2, R3,..., , в каждом из которых может быть записано натуральное число. Часть этих регистров используют для хранения команд (аналог левой полуленты), остальные - для хранения данных, промежуточных вычислений и окончательных результатов (аналог правой полуленты). Модель позволяет организовать произвольный доступ к ячейкам памяти. Такую модель назвали «машиной произвольного доступа» - МПД.

Набор команд МПД включает:

- **команду обнуления**: действие команды заключается в замене содержимого регистра R_n на 0, т.е. $R_n := 0$,
- **команду прибавления единицы**: действие команды заключается в увеличении содержимого регистра R_n на 1, т.е. $R_n := R_n + 1$,
- **команду переадресации**: действие команды заключается в замене содержимого регистра R_n числом r_m, хранящимся в регистре R_m, т.е. $R_n := R_m$,
- **команду условного перехода**: действие команды заключается в сравнении содержимого регистров R_n и R_m:

  - если $R_n = R_m$, то перейти к исполнению команды q_i;
  - если $R_n \neq R_m$, то перейти к исполнению команды q_j.

Упорядоченная последовательность команд формирует программу МПД.

3. Нормальный алгоритм Маркова - модель алгоритма

Понятие "нормальный алгоритм" ввел в 1947 г. советский ученый А.А. Марков в качестве одного из уточнений представления об алгоритме. Он положил, что ***нормальный алгоритм***, являясь алгоритмом в некотором алфавите V_T, порождает в нем некоторый детерминированный процесс переработки только одного слова P_o и только в одном алфавите.

Словами P_i в алгоритме Маркова могут быть арифметические, алгебраические или логические выражения.

Нормальный алгоритм Маркова есть указание использовать ***упорядоченный список*** правил подстановки:

$$\alpha_i \Rightarrow \beta_i,$$

где α_i и β_i — некоторые слова в алфавите V_T.

Множество правил и порядок их использования позволяют выполнять преобразования исходного слова P_o в заключительное слово Q, т. е.

$$P_o \rightarrow P_1 \rightarrow P_2 \rightarrow ... \rightarrow P_i \rightarrow ... \rightarrow Q.$$

Для организации последовательного и упорядоченного просмотра правил они должны быть индексированы $i \in \{1, 2, 3,...\}$.

Если слово P_i есть цепочка вида $(\gamma_1\alpha_i\gamma_2)$ в алфавите $V_т$, где γ_1 и γ_2 – слова в этом же алфавите и среди множества правил первым в упорядоченном списке есть правило $\alpha_i \Rightarrow \beta_i$, то нужно выполнить подстановку $(\gamma_1\alpha_i\gamma_2) \Rightarrow (\gamma_1\beta_i\gamma_2)$.

Суть упорядоченного использования правил состоит в том, что каждое переработанное слово вновь поступает в «начало» алгоритма и вновь проверяется на подстановку правил в соответствии с протоколом.

Среди множества правил выделяют заключительное $\alpha_i \rightarrow \bullet \beta_i$, результатом подстановки которого формируется слово Q и дается указание об окончании работы алгоритма.

Процесс может оборваться на некотором слове P_i, для которого нет соответствующего правил. Тогда это слово направляется в «тупик».

Для того чтобы построить модель алгоритма, необходимо выделить упорядоченную последовательность левых частей правил подстановки, так называемых ***распознавателей вхождения*** слов α_i в слово P_i, и множество соответствующих ***операторов подстановки*** слова β_i в слово P_{i+1}.

На схеме алгоритма (см. рис. 19) эти блоки обозначены так:

- распознаватели вхождения - $РВ_i$;
- операторы подстановки - $ОП_i$.

Распознаватели вхождения соединяются последовательно в соответствии с заданной последовательностью правил. Второй выход распознавателя вхождения при обнаружении α_i в слове P_i передает информацию о слове $P_i=\gamma_1\alpha_i\gamma_2$ в $ОП_i$, где выполняется соответствующая замена слова α_i на слово β_i, т. е.
$\gamma_1\alpha_i\gamma_2 \Rightarrow \gamma_1\beta_i\gamma_2=P_{i+1}$.

Оператор подстановки направляет слово P_{i+1} в «начало» алгоритма, если применена простая подстановка, и в «конец» алгоритма, если применена заключительная подстановка.

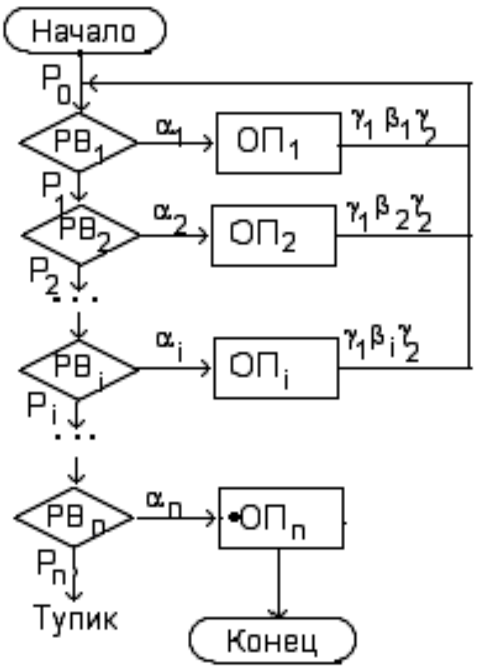


Рис. 19. Схема нормального алгоритма Маркова

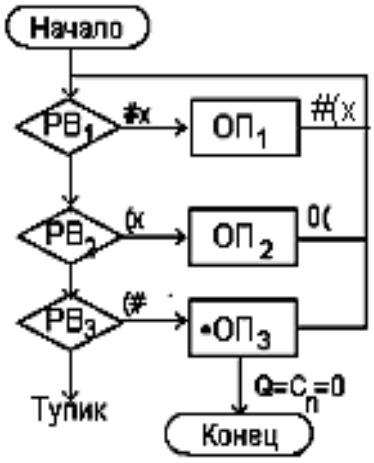
Ниже дано несколько примеров вычисления базовых и примитивно-рекурсивных функций по схеме нормального алгоритма Маркова.

3.1 Примеры нормальных алгоритмов Маркова

Пример 36**.** Вычислить базовую функцию $C_n(x)=0$ в десятичной системе счисления по схеме нормального алгоритма Маркова.

Пусть «# XXX... #» - число в десятичной системе счисления, X – цифра1, 2, 3,… Алфавит V_T содержит четыре символа v_T = {X, 0, #, (},

На рис. 20 приведена схема этого алгоритма.



Протокол:

1] #X → #(X,

2] (X → 0(,

3] (# → •#.

Пусть x=#289#. Тогда P_0=#289#⇒#(289# ⇒#0(89#⇒#00(9#⇒#000(#⇒#000•# = #000#=Q.

Рис. 20. Схема алгоритма вычисления Cn = 0.

Пример 37. Вычислить базовую функцию λ(x) в десятичной системе счисления.

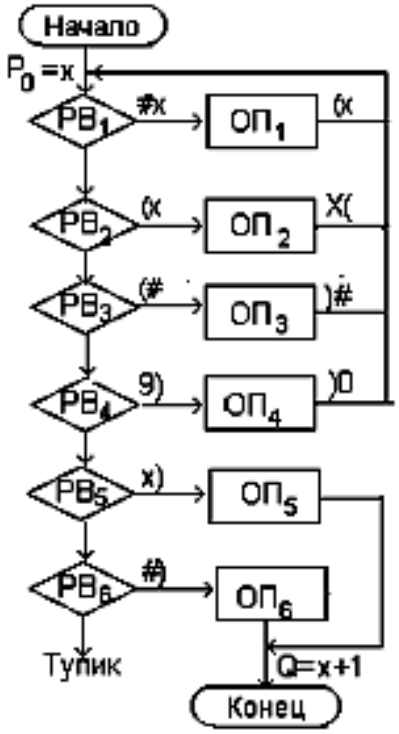


Рис. 21. Схема λ(x)

Пусть *x* =«# XXX ... #» - число в десятичной системе счисления, X – цифра. Алфавит содержит четыре символа V_T={X, #, (,)}.

Протокол.

1] #X⇒#(X,

2] (X⇒X(,

3] (#⇒)#,

4] 9) ⇒)0,

5] X) ⇒• X+1,

6] #) ⇒•1.

Схема алгоритма приведена на рис. 21.

Пусть x=0. Тогда P_0=#0#⇒#(0#⇒#0(#⇒#0)#⇒ ⇒•1=1=Q.

Пусть x=289. Тогда P_0=#289#⇒#(289#⇒ ⇒#2(89#⇒#28(9#⇒#289(#⇒#289)#⇒ #28)0#⇒ ⇒#2•90#⇒#290#=Q.

Пусть x=999. Тогда P_0=#999#⇒#(999#⇒ ⇒#9(99#⇒#99(9#⇒#999(#⇒#999)#⇒#99)0#⇒# ⇒9)00#⇒#)000#⇒ ⇒#•1000#=#1000#=Q.

Пример 38. Преобразовать цифры десятичного числа в унарный код.

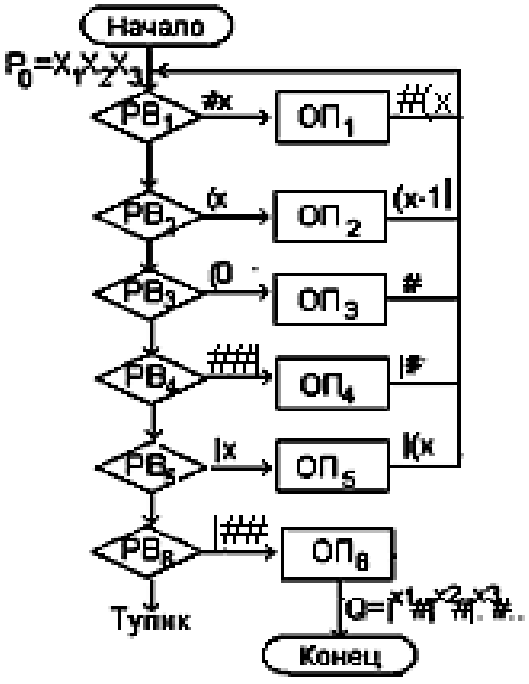


Рис.22. Схема алгоритма преобразования числа в унарный код.

Если P_o=#$X_1X_2X_3$...#, где X_1,X_2,X_3,… - цифры, то Q=#|^{x1}#|^{x2}#|^{x3}#... , где |^{xI} – x_i «палочек» унарного кода. Алфавит V_T для преобразования цифры в код содержит символы V_T = {X, #, |, (}. На рис. 22. приведена схема этого алгоритма.

Протокол.

1] # X →#(X,

2] (X →(X-1|,

3] (0 → #,

4] ##|→#|,

5] |X →|(X,

6] |## →•|#.

Пусть x=234. Тогда P_0=# 234##⇒#(234##⇒#(1|34##⇒#(0||34##⇒##||34##⇒
⇒#||34##⇒#||(34##⇒#||(2|4##⇒ #||(1||4##⇒#||(0|||4##⇒#||##|||4##⇒#||#|||(4##⇒
⇒#||#|||(3|#|⇒#||#|||(2||##⇒#||#|||(1|||##⇒#||#|||(0||||##⇒#||#|||##||||##⇒ #||#|||#||||##⇒
⇒#||#|||#|||•|## =#||#|||#||||## Q.

Пример 39. Вычислить сумму двух чисел в унарном коде,
т.е. f (x; y) = $|^{x+y}$. Пусть P_o =#$|^x$+$|^y$#. Тогда Q =# $|^{x+y}$#.
Алфавит V_T= { |, +, #, (}.

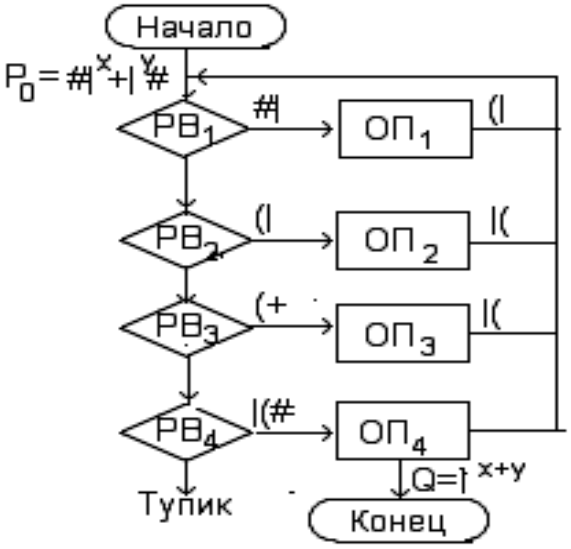


Рис.23 Схема алгоритма
сложения двух чисел

Протокол .
1] #| → #(|,
2] (| → |(,
3] (+ → |(,
4] |(# → •# (см. рис. 23).

Пусть x=2, y=3.
Тогда P_o = #||+|||#⇒ #(||+|||#⇒
#|(|+|||#⇒ #||(+|||#⇒ #|||(|||#⇒ #||||(||#⇒
#|||||(|#⇒ #||||||(#⇒#|||||•#=#|||||#=Q.

Пример 40. Вычислить произведения двух чисел в унарном коде: f(x; y)=$|^{x\cdot y}$.

Пусть P_o= #$|^x \cdot |^y$ #, тогда Q = #$|^{x\cdot y}$ #.
Алфавит $V_т$={| , ·, #, (,)}.

Протокол.
1] ·| | →(·|, 6] (| →(,
2] ·|# →(#, 7] ( ) →),
3] |(→(|), 8]) →|,
4])(→( ) 9] |# →•|#
5])| →|),

Схема алгоритма приведена на рис. 24.
Пусть x = 3 и y = 2
Тогда P_0=#|||∗||#⇒#|||(∗|#⇒#||(|)∗|#⇒ #|(|)|)∗|#⇒#(|)|)|)∗|#⇒#()|)|)∗|#⇒#)|)|)∗|#⇒
#|))|)∗|#⇒#|)|))∗|#⇒#||)))∗|#⇒#||)))(#⇒#||))()#⇒
#||)())#⇒#||()))#⇒#|(|))))#⇒#(|)|))))#⇒#()|))))#⇒
#)|))))#⇒#|)))))#⇒#||))))#⇒#|||)))#⇒#||||))#⇒ #|||||)#⇒#||||||#⇒#|||||•|#=Q.

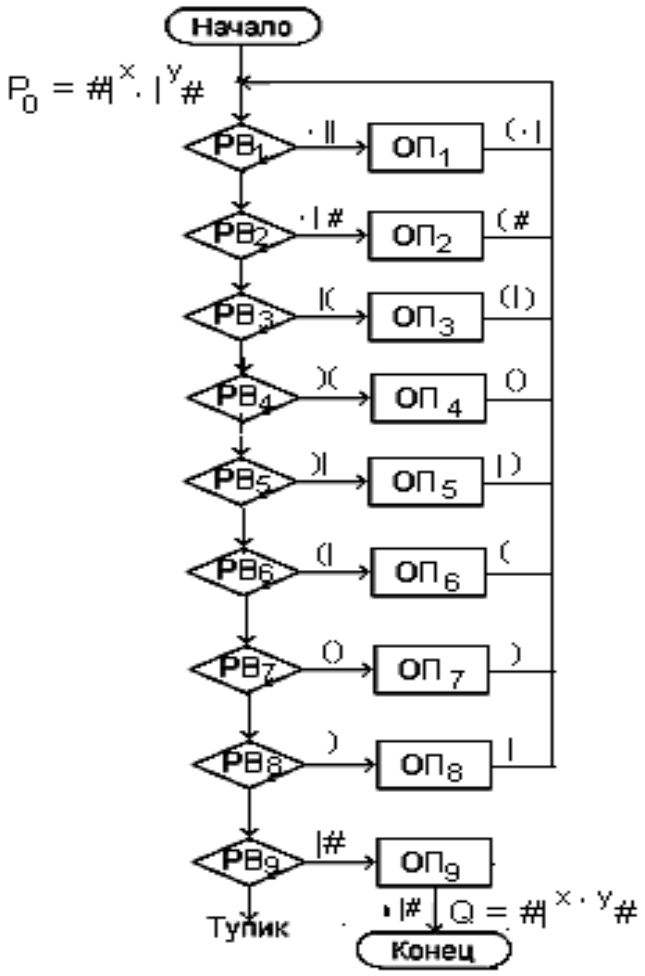


Рис. 24. Схема алгоритма умножения чисел

3.2. Нумерация слов

Пусть дан алфавит A = {a_1, a_2, …, a_r}, где r – число букв или символов алфавита. Например, для латиницы это - r=26. На множестве букв алфавита согласно правилам соответствующей грамматики можно сформировать множество слов конечной длины A*={α_1, α_2, …, β_1, β_2,…, ε}, где ε -пустое слово. Согласно правилам алгоритма Маркова это могут быть слова α_i и β_i. Тогда алфавитная нумерация определяется как:

$$\begin{cases} h(\varepsilon)=0, \\ h(a_{i_1}, a_{i_2},..., a_{i_s}) = i_s + i_{s-1} \cdot r + i_{s-2} \cdot r^2 + ... + i_1 \cdot r^{s-1}. \end{cases}$$

Поскольку при фиксированном r каждое положительное число n однозначно представимо в виде:

$n=i_s + i_{s-1} \cdot r + \ldots + i_1 \cdot r^{s-1}$, где $1 \le i \le r$ – порядковый номер символа в алфавите A, нижний индекс которого указывает его место в слове α_i или β_i.

Например, для нормального алгоритма Маркова произведения двух чисел (см. прим. 40) алфавит содержит пять символов $V_т$={| , ·, #, (,)}, а протокол правила α_1:= ·| | → β_1:= (·|, α_2:= ·|# → β_2:= (#, α_3:= |(→ β_3:= (|), α_4:=)(→ β_4:= ( ), α_5:=)| → β_5:= |), α_6:= (| → β_6:= (, α_7:= ( ) → β_7:=), α_8:=) → β_8:= |, α_9:= |# → β9:= •|#.

Номера некоторых слов: α_1:= ·| | : n = 1+1·5+2·5^2 = 56,
β_1:= (·| : n = 1+2·5+4·5^2 =111, α_2:= ·|# : n = 3+1·5+2·5^2 = 58, β_2:= (# : n = 3+4·5 = 23, α_3:= |(: n = 4+1·5 = 9, β_3:= (|): n = 5+1·5+4·5^2 = 110, α_4:=)(: n = 4+5·5 = 29. β_4:= ( ) : n = 5+4·5 =25 и т.д.

Следовательно, цифровой код протокола есть (56→111, 58→23, 9→110, 29→25, 26→10, 21→4, 25→5, 5→1, 8→8). Далее нужно найти номер для набора двух чисел команды (см. 1.3). Так может быть сформирована упорядоченная последовательность номеров команд, моделирующая заданный алгоритм.

4. Вычислимость и разрешимость

Выше было доказано, что классы функций, вычислимых с помощью рекурсивных функций, машин Тьюринга или нормальных алгоритмов Маркова, совпадают. Это позволяет рассматривать понятие «вычислительный алгоритм» инвариантным к способу описания. Различия наблюдаются лишь в использовании алгоритмических объектов. Если для рекурсивных функций объектами являются числа и числовые функции, а процесс вычисления задан операторами суперпозиции, рекурсии, минимизации и итерации, то для машин Тьюринга такими объектами являются символы алфавитов внешней и внутренней памяти, а процесс вычисления задан протоколом, использующим функции выхода, перехода и перемещения головки. Для нормального алгоритма Маркова такими объектами являются слова или последовательности символов, а процесс вычисления задан правилами подстановки или ***продукциями***, изменяющими состав и структуру исходной последовательности символов до искомого результата.

Функция называется вычислимой, если существует вычисляющий её алгоритм. «***Вычислимость***[*]» является одним из основных понятий теории алгоритмов, инвариантном к вычисляемой функции и алгоритму. Различие между вычислимой функцией и алгоритмом – это различие между описанием функции и способом вычисления её значений при заданных значениях независимых аргументов.

Если для решения задачи, принадлежащей единому классу задач, найден алгоритм вычисления, то о задаче говорят как об алгоритмически разрешимой проблеме. Иначе говоря, обязательным условием вычислимости или результативности вычисления является её алгоритмическая разрешимость. В этом смысле понятие «***разрешимость***[**]» является также основным понятием в теории алгоритмов.

Анализ трех типов моделей показывает, что основные свойства дискретности, детерминизма, массовости и результативности остаются неизменными для различных способов описания:

- ***Свойство дискретности***: алгоритм состоит из отдельных элементарных действий, выполняемых по шагам; множество элементарных шагов, из которых состоит алгоритмический процесс, конечно и счетно.
- ***Свойство детерминированности***: после каждого шага дается точное указание, как и в какой последовательности выполнять следующие шаги алгоритмического процесса.
- ***Свойство массовости***: использование алгоритма допустимо для множества алгоритмических объектов данного типа и данного класса задач.
- ***Свойство результативности***: остановка алгоритмического процесса обязательна после конечного числа шагов с указанием искомого результата.

---

[*] см. [2], [4], [5]

5. Сложность вычислений*

Выбор алгоритмической модели существенно влияет на сложность вычисления задачи. ***Сложность вычисления*** есть функция, дающая числовую оценку трудоемкости применения алгоритма к исходным данным для получения искомого результата. Чаще всего рассматривают ***временну́ю сложность*** и ***ёмкостну́ю***, которые чаще всего представляются функциями от |x|, где |x| – мощность множества исходных данных.

Время вычисления описывается произведением числа шагов алгоритма от исходных данных до искомого результата н средним физическим временем реализации одного шага. Число шагов определяется его описанием в заданной алгоритмической модели. Пусть машина Тьюринга вычисляет некоторую функцию f(x) данного класса задач. Тогда t(x) есть функция, равная числу шагов при вычислении f(x), если f(x) определена. Функция t(x) называется ***временно́й сложностью***. Однако физическое время реализации одного шага алгоритма на конкретном компьютере зависит от типа компьютера, способов компиляции, скорости обработки информации, что существенно усложняет определение временно́й сложности.

Объем памяти, как количественная характеристика алгоритма, определяется количеством единиц памяти, используемых в процессе вычисления алгоритма. Эта величина не может превосходить максимального числа единиц памяти, используемых на одном шаге алгоритма. Пусть машина Тьюринга вычисляет функцию f(x). Тогда s(x) есть функция, равная множеству всех ячеек информационной ленты, которые, если f(x) определена, посещаются в процессе вычисления этой функции. Функция s(x) называется ***ёмкостно́й сложностью***.

Если машина Тьюринга имеет внешнюю и внутреннюю памяти мощности $n=|V_t|$ и $m=|Q|$ соответственно, то для временно́й и ёмкостно́й функций сложности допустимы оценки:

$$\begin{cases} s(x) \le |x| + t(x), \\ t(x) \le m \cdot s^2(x) \cdot n^{s(x)}. \end{cases}$$

Чаще всего временну́ю сложность t(x) описывают полиномами от исходных данных. Если известен размер исходных данных - |x| и временная сложность задана некоторым полиномом t(|x|)=p(|x|), то ***оценка временно́й сложности*** есть O(p(|x|)). Эта оценка определяется, как правило, старшим членом полиномиального ряда. При этом говорят, что машина Тьюринга решает задачу за ***полиномиальное время***. Например, если временна́я сложность задана полиномом $p(|x|)\_=4|x|^2+7|x|+12$, то оценка временной сложности есть $O(|x|^2)$. Алгоритм, имеющий полиномиальное время, называют ***полиномиальным алгоритмом.*** Множество однотипных задач, разрешаемых на машине Тьюринга за полиномиальное время, принадлежит классу задач P.

Есть задачи, для которых может быть найдено некоторое слово α , являющееся «***догадкой***» (или ***удостоверением***) принадлежности задачи к классу P на

* см. [2], [4], [5]

недетерминированной машине Тьюринга. В этом случае говорят, что задача принадлежит к классу NP, а оценка временной сложности есть $O(p(\alpha))$.

Если $t(|x|)\geq O(p(|x|))$, то говорят, что машина Тьюринга решает задачу за ***экспоненциальное время*** $t(|x|) = O(c^{|x|})$, где c – некоторая константа. Например, для булевых функций c=2. Тогда $t(n) = O(2^n)$, где n =число булевых переменных.

Пример 41. Дано $N=\{1, 2, 3,\ldots, n\}$ и $R\subseteq N\otimes N$. Является ли R отношением эквивалентности?

Известно, что R является отношением эквивалентности тогда и только тогда, когда выполнены условия:

$$r(i,j)=\begin{cases}1, \text{ если } (i,\ j)\in R \text{ и } (j,\ i)\in R,\\ 1, \text{ если } i=j,\\ 0, \text{ если } (i,\ j)\notin R.\end{cases}$$

Дополнительным условием является $r^2(i,j)\geq r(i,j)$.

Проверка этих условий ограничена независимыми оценками $O(n^2)$ и $O(n)$, так как нужно вычислять значения $r^2(i, j)$ для каждой пары (i, j), а затем сравнивать с r(i , j). Оценка временно́й сложности задачи есть $O(n\cdot n^2) = O(n^3)$. Поэтому алгоритм данной задачи есть полиномиальный, а оценка его временно́й сложности зависит от числа исходных данных в третьей степени.

Пример 42. Дано $N=\{1, 2, 3,\ldots, n\}$ и $R=\{(1, 2), (2, 3), (3, 4),\ldots, (n, 1)\}$ $\subseteq N\otimes N$. Является ли R отношением эйлеровым?

Бинарное отношение R называют эйлеровым, если элементы R можно упорядочить $(i_1, i_1), (i_2, i_2),\ldots, (i_n, i_n)$, где $n=|R|$, и выполнить условие $(i_1, i_2)\in R$, $(i_2, i_3)\in R,\ldots, (i_n, i_1)\in R$.

Связное отношение R является эйлеровым тогда и только тогда, когда число единиц в матрице R совпадает в i-м столбе и i-й строке для каждого $i\in\{1,2..,n\}$.

Так как все вычисления выполняются только с одной матрицей R, сравнивая число единиц в i-й строке и i-м столбце, то временна́я сложность задачи будет $O(n^2)$. Поэтому алгоритм данной задачи полиномиальный, а оценка его временно́й сложности зависит от числа исходных данных во второй степени.

Пример 43. Дана формула $f(x_1, x_2,..., x_n) = \&(\overline{x_1^{\sigma_1}} \vee \overline{x_2^{\sigma_2}} \vee \ldots \vee \overline{x_n^{\sigma_n}})$, где $\sigma_i\in\{0,1\}$ – значение x_i булевой переменной. Проверить её выполнимость.

Алгоритм проверки выполнимости этой формулы требует перебора 2^n наборов $(\sigma_1, \sigma_2,..., \sigma_n)$ и вычисления значения функции для каждого набора. Поэтому $t(|x|) = O(2^n)$, т. е. алгоритм данной задачи - экспоненциальный.

Пример 44. Дано $N=\{1, 2, 3,\ldots, n\}$ и $R=\{(1, 2), (2, 3), (3, 4),\ldots, (n, 1)\} \subseteq N \otimes N$. Является ли R отношением гамильтоновым?

Бинарное отношение называют гамильтоновым, если элементы множества N можно упорядочить так $(i_1, i_2,\ldots, i_n)$, чтобы выполнялось условие: $(i_1, i_2) \in R$, $(i_2, i_3) \in R,\ldots, (i_n, i_1) \in R$.

Догадкой (или удостоверением) гамильтонова отношения R является заданная последовательность $\alpha(R)=(i_1, i_2,\ldots, i_n)$. Проверяя бинарное отношение $R=\{(1, 2), (2, 3), (3, 4),\ldots, (n, 1)\}$ по заданной последовательности $\alpha=(1, 2,\ldots, n)$, можно убедиться, что задача проверки отношения R находится в классе NP задач.

Вопросы и задачи

1. Какие функции могут быть получены по схеме примитивной рекурсии:

a) $g(x)= J_{1,1}$, $h(x, y, f(x, y))= f_{+}(J_{3,1}, J_{3,3})$,

b) $g(x)= J_{1,1}$, $h(x, y, f(x, y))= f_{+}(J_{3,1}, J_{3,2})$,

c) $g(x)= J_{1,1}$, $h(x, y, f(x, y))= f_{+}(J_{3,2}, J_{3,3})$,

d) $g(x)=\lambda(J_{1,1})$, $h(x, y, f(x, y)) = f_{+}(J_{3,1}, J_{3,3})$,

e) $g(x) = C_1 = 0$, $h(x, y, f(x, y)) = \lambda(J_{3,3})$.

f) $g(x)=\lambda(J_{1,1})=x+1$ $h(x, y, f(x, y)) = f_{+}(J_{3,1}, J_{3,3})=x+f(x,y)$.

2. Применить оператор минимизации по переменной y. Найти $\varphi(x)$.

a) $f(a, y) = \left[\frac{y}{a}\right]$,

b) $f(x, y) = (x - 2\cdot y^2)$,

c) $f(x, y) = (x^2 - 2\cdot y)$,

d) $f(x, y)=\left[x \dot{-} 2 \cdot y\right]$,

e) $f(x, y) = [(x^2 \dot{-} 3 \cdot y)]$,

3. Доказать тождества.

a) $x+(y \dot{-} x) = y+(x \dot{-} y)$,

b) $x \dot{-} (y+z) = (x \dot{-} y) \dot{-} z$,

c) $(x \dot{-} y) \dot{-} z = (x \dot{-} z) \dot{-} y$.

4. Какую функцию f(x) можно вычислить, используя оператор итерации $J^n(h)$ и функцию h:

a) $h(f(i) = (f(i) + 1 + \left[\sqrt{4 \cdot f(i) + 1}\right])$,

b) $h(f(i)= (f(i)+2[\sqrt{f(i)}]$,

5. Какую функцию вычисляет машина Тьюринга:

$q_o | \rightarrow q_1 \# П$,

$q_1 | \rightarrow q_1 | П$,

$q_1 \# \rightarrow q_2 | Л$,

$q_2 | \rightarrow q_2 | Л$,

$q_2 \# \rightarrow q_0 | С$. Показать процесс на информационной ленте.

6. Написать протоколы, построить таблицу поведения, нарисовать граф поведения машины Тьюринга:

а) сдвиг слова влево на одну ячейку:

$\#q_0|^x \# \rightarrow q_k|^x \#\#$,

b) перенос слова с правой на левую полуленты:

$\#q_0|^x\# \rightarrow \#|^{x-1}q_k|\#$,

c) удвоение слова:

$\#|^{x-1}q_0|\# \rightarrow \#|^{2x-1}q_k|\#$.

Ответы и решения

1a) $f(x, 0) = x$,
$f(x, 1) = x+f(x, 0) = x+x = 2x$,
$f(x, 2) = x+f(x, 1) = x+2x = 3x$,
$f(x, 3) = x+f(x, 2) = x+3x = 4x$,
……
$f(x, i) = x+f(x, i-1) = x+i\cdot x = x\cdot(i+1)$.
Если $i = y$, то $f(x, y) = x\cdot y+x$.

1b) $f(x,0) = x$,
$f(x, 1) = x+0 = x$,
$f(x, 2) = x+1 = x+1$.
$f(x, 3) = x+2 = x+2$,
……
$f(x, i) = x+(i-1) = x+(i -1)$.
Если $i = y$, то $f(x, y) = (x+y)-1$.

1c) $f(x, 0) = x$,
$f(x, 1) = 0+f(x, 0) = x$,
$f(x, 2) = 1+ f(x, 1) = 1+x$,
$f(x, 3) = 2+ f(x, 2) = 3+x$,
……
$f(x, i) = (i+x)$.
Если $i = y$, то $f(x, y) = (y+x)$.

1d) $f(x, 0) = \lambda(x) = x+1$,
$f(x, 1) = x+ f(x, 0) = x+x+1 = 2\cdot x+1$,
$f(x, 2) = x+ f(x, 1) = x+2\cdot x+1 = 3\cdot x+1$,
$f(x, 3) = x+ f(x, 2) = x+3\cdot x+1 = 4\cdot x+1$,
……
$f(x, i) = (i+1)\cdot x+1$.
Если $i = y$, то $f(x, y) = y\cdot x+x+1$.

1e) $f(x, 0) = 0$,
$f(x, 1) = (f(x, 0)+1 = 1$,
$f(x, 2) = f(x, 1)+1= 1+1 = 2$,
$f(x, 3) = f(x, 2)+1= 2+1 = 3$,
……
$f(x, i) = i$.
Если $i = y$, то $f(x, y) = y$.
Если $g(x)=\lambda(J_{1,1})=x+1$ и $h(x, y, f(x, y)) = f_{+}(J_{3,1}, J_{3,3})=x+f(x,y)$.

По схеме примитивной рекурсии:

1f) f (x, 0)=g(x)=(x+1),
f(x, (0+1))=x+(x+1)=(2x+1),
f(x, (1+1))=x+(2x+1)=(3x+1),
f(x, (2+1))=x+(3x+1)=(4x+1),
f(x, (3+1))=x+(4x+1)=(5x+1),
f(x, (4+1))=x+(5x+1)=(6x+1),
…….
f(x, (i+1))=x+((i+1)·x+1)=(x·((i+1)+1)+1.

Следовательно, если (i+1)=y, то f(x, y)=x·λ(y)+1=x·y+x+1 есть рекурсивная функция, так как для ее вычисления использованы базовые функции тождества и следования и примитивно рекурсивная функция сложения.

2a) $f(a, y) = \left[\frac{y}{a}\right]$ при a=4.

Согласно алгоритму следует вычислить $\varphi(a) = \mu_y(f(a,y)\cdot\overline{Sg}(y)) = 0)$.

$$\mu_0(\left[\frac{0}{4}\right]\cdot\overline{Sg}(0)\neq 0),\ \mu_1(\left[\frac{1}{4}\right]\cdot\overline{Sg}(1)\neq 0),\ \mu_2(\left[\frac{2}{4}\right]\cdot\overline{Sg}(2)\neq 0),$$
$$\mu_3(\left[\frac{3}{4}\right]\cdot\overline{Sg}(3)\neq 0),\ \mu_4(\left[\frac{4}{4}\right]\cdot\overline{Sg}(4)=0).$$

Следовательно, $\varphi(4) = \mu_4(f(4,4)\cdot\overline{Sg}(4)=0) = 4$.

2b) $f(x, y) = (x - 2\cdot y^2)$.

Пусть x=8. Тогда $\mu_0((8\text{-}2\cdot 0^2)\neq 0)$, $\mu_1((8\text{-}2\cdot 1^2)\neq 0)$, $\mu_2((8\text{-}2\cdot 2^2)=0)$.

Следовательно, $\varphi(8)=\mu_2((8\text{-}2\cdot 2^2)=0)=2$.

Область определения функции φ(x) есть x= 2, 8, 18, 32,…, а область значений y=1, 2, 3, 4,…, т.е. φ(x) есть частично рекурсивная функция.

2c) $f(x, y) = (x^2 - 2\cdot y)$.

Пусть x=2. Тогда $\mu_0(2^2 - 2\cdot 0\neq 0)$, $\mu_1(2^2 - 2\cdot 1\neq 0)$, $\mu_2(2^2 - 2\cdot 2=0)$.

Следовательно, $\varphi(2) = \mu_2(2^2 - 2\cdot 2=0)=2$.

2d) для x=5 имеем f(5, 0)=5≠0, f(5, 1)=3≠0, f(5, 2)=1≠0, f(5, 3)=0,
т.е. φ(5) =3.

2e) для x=1 f(1, 0)=1≠0, f(1, 1)=0, т.е. φ(1)=1,
для x=2 f(2, 0)=4≠0, f(2, 1)=1≠0, f(2, 2)=0, т.е. φ(2)=2,
для x=3 f(3, 0)=9≠0, f(3, 1)=6≠0, f(3, 2)=3 f(3, 3)=0, т.е. φ(3)=3,
для x=4 имеем f(4, 0)=16≠0, f(4, 1)=13≠0, f(4, 2)=10≠0, f(4,3)=7≠0,
f(4, 4)=4≠0, f(4, 5)=1≠0, f(4, 6)=0, т.е. φ(4)=6 и т.д.

Область определения функции φ(x)=y задана на множестве целых чисел x={1, 2, 3,…}, а область её значений также на множестве целых чисел $y\geq [x^2/3]$.

3a) $x+(y\dot{-}x) = y+(x\dot{-}y)$.

$$x+(y\dot{-}x)=\begin{cases} x+y-x=y, & \text{если } x<y, \\ x+0=x, & \text{если } x\ge y \end{cases} \quad \text{и} \quad y+(x\dot{-}y)=\begin{cases} y+x-y=x, & \text{если } y<x, \\ y+0=y, & \text{если } y\ge x. \end{cases}$$

Следовательно, если x>y, то $x+(y\dot{-}x) = y+(x\dot{-}y) = x$, если x<y, то $x+(y\dot{-}x) = y+(x\dot{-}y) = y$, если x = y, то $x+(y\dot{-}x) = y+(x\dot{-}y) = y = x$, *ч.т.д.*

3b) $x\dot{-}(y+z) = (x\dot{-}y)\dot{-}z$,

$$x\dot{-}(y+z)=\begin{cases} x-y-z), & \text{если } x>(y+z), \\ 0, & \text{если } x\le(y+z), \end{cases}$$

$$(x\dot{-}y)\dot{-}z=\begin{cases} x-y-z, & \text{если } (x-y)>z \text{ или } x>(y+z), \\ 0, & \text{если } x\le(y+z). \end{cases}$$

Следовательно, если x>(y+z), то $x\dot{-}(y+z) = (x\dot{-}y)\dot{-}z = (x-y-z)$, если x<(y+z), то $x\dot{-}(y+z) =(x\dot{-}y)\dot{-}z = 0$, если x = y, то $x\dot{-}(y+z) = (x\dot{-}y)\dot{-}z=0$, *ч.т.д.*

3c) $(x\dot{-}y)\dot{-}z = (x\dot{-}z)\dot{-}y$.

$$(x\dot{-}y)\dot{-}z=\begin{cases} x-y-z, & \text{если } x>(y+z), \\ 0, & \text{если } x\le(y+z), \end{cases} \qquad (x\dot{-}z)\dot{-}y=\begin{cases} x-y-z, & \text{если } x>(y+z), \\ 0, & \text{если } x\le(y+z). \end{cases}$$

Следовательно, если x>(y+z), то $(x\dot{-}y)\dot{-}z = (x\dot{-}z)\dot{-}y = (x-y-z)$, если $x\le(y+z)$, то $(x\dot{-}y)\dot{-}z = (x\dot{-}z)\dot{-}y = 0$, *ч.т.д.*

4a) $h(f(i)=(f(i)+1+\left[\sqrt{4\cdot f(i)+1}\right])$.

f(0)=0,

f(1)=h(f(0))=(0+1+[$\sqrt{4\cdot 0+1}$])=1+1=2,

f(2)=h(f(1))=(2+1+[$\sqrt{4\cdot 2+1}$])=3+3=6,

f(3)=h(f(2))=(6+1+[$\sqrt{4\cdot 6+1}$])=7+5=12,

f(4)=h(f(3))=(12+1+[$\sqrt{4\cdot 12+1}$])=13+7=20,

……

$f(i)=h(f(i-1))=i^2+i$.

Если i=x, то $f(x)=x^2+x$.

4b) h(f(i)= (f(i)+1+2·[$\sqrt{f(i)}$].

f(0)=0,

f(1)=(f(0)+1+2·[$\sqrt{f(0)}$]=1,

f(2)=(f(1)+1+2·[$\sqrt{f(1)}$]=4,

f(3)=(f(2)+1+2·[$\sqrt{f(2)}$]=9,

f(4)=(f(3)+1+2·[$\sqrt{f(3)}$]=16,

……

$f(i)=i^2$.
Если i=x, то $f(x)=x^2$.

5. Пусть x=3, т.е. на информационной ленте слово # | | | #.

| | | # → # # | | # → # # | | # → # # | | # # ← # # | | | #← # # | | | # ← # # | | |#←
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
q_0 q_1 q_1 q_1 q_2 q_2 q_2 q_2
| | | |#
↑
q_0 Следовательно, машина Тьюринга вычисляет функцию λ(x).

6a) T_1: $\#q_0|^x\# \rightarrow q_k|^x\#\#$,

Протокол:
$q_0|\rightarrow q_1|П$,
$q_1|\rightarrow q_1|П$,
$q_1\#\rightarrow q_2\#Л$,
$q_2|\rightarrow q_3\ \#Л$,
$q_3|\rightarrow q_3\ |Л$,
$q_3\#\rightarrow q_k|С$

$q_i \in Q$	V_t	
	|	#
q_0	q_1|П	-
q_1	q_1|П	q_2#Л
q_2	q_3#Л	-
q_3	q_3|Л	q_k|С

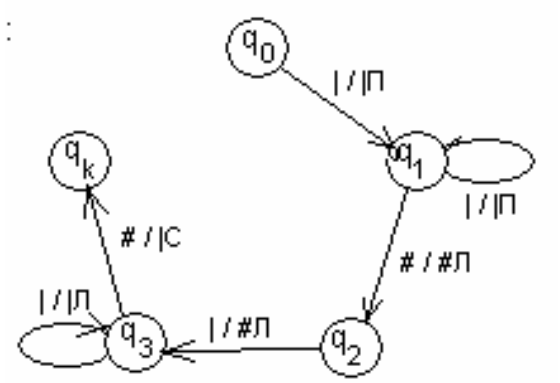


Рис. 0.1. Граф алгоритма

6b) T_2: $\#q_0|^x\# \rightarrow \#|^{x-1}q_k|\#$,

Протокол:
$q_0| \rightarrow q_1|П$,
$q_1| \rightarrow q_1|П$,
$q_1\# \rightarrow q_2\#Л$,
$q_2| \rightarrow q_k|С$.

$q_i \in Q$	V_t	
	|	#
q_0	q_1|П	-
q_1	q_1|П	q_2#Л
q_2	q_k|С	-

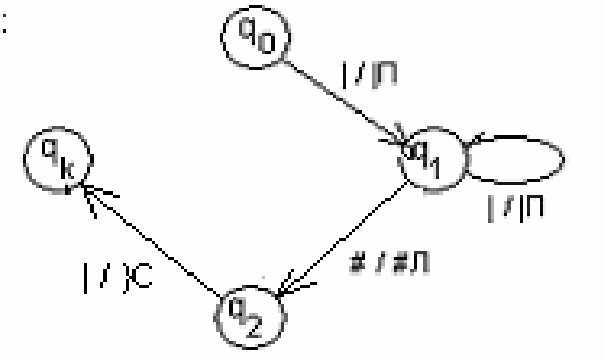


Рис. 0.2. Граф алгоритма

6c) T_3: $\#|^{x-1}q_0|\# \rightarrow \#|^{2x-1}q_k|\#$

Протокол:

$q_0|\rightarrow q_1)Л$, $q_4\ \#\rightarrow q_7\#Л$,
$q_1|\rightarrow q_1|Л$, $q_4(\rightarrow q_4|П$,
$q_1\#\rightarrow q_2(Л$, $q_4)\rightarrow q_4|П$,
$q_1(\rightarrow q_5(Л$, $q_5|\rightarrow q_5|Л$,
$q_2\#\rightarrow q_3|П$, $q_5\#\rightarrow q_6|П$,
$q_3(\rightarrow q_3(П$, $q_6|\rightarrow q_3|П$,
$q_3|\rightarrow q_3|П$, $q_7|\rightarrow q_8\#Л$,
$q_3)\rightarrow q_4)Л$, $q_8|\rightarrow q_k|С$.
$q_4|\rightarrow q_1)Л$,

$q_i \in Q$	V_t			
	|	#	(	)
q_0	$q_1)Л$	-	-	-
q_1	$q_1\|Л$	$q_2(Л$	$q_5(Л$	-
q_2	-	$q_3\|П$	-	-
q_3	$q_3\|П$	-	$q_3(П$	$q_4)Л$
q_4	$q_1)Л$	$q_7\#Л$	$q_4\|П$	$q_4\|П$
q_5	$q_5\|Л$	$q_6\|П$		
q_6	$q_3\|П$			
q_7	$q_8\#Л$			
q_8	$q_k\|С$			

Индивидуальное задание

1. Выделить машины Тьюринга, реализующие служебные и вычислительные примитивно рекурсивные функции.
2. Для каждой машины Тьюринга составить таблицу поведения и нарисовать граф.
3. Выполнить композицию машин Тьюринга, составив схему соединения и обобщенную таблицу поведения.
4. Проверить “усеченное вычитание” для x>y>z>s>t и x≤y≤z≤s≤t.

Вариант	Задание	Вариант	Задание	Вариант	Задание	Вариант	Задание
	$q_0\|^{x}\#\|^{y}\#\|^{z}\#\|^{s}\#\|^{t}\#\rightarrow$		$q_0\|^{x}\#\|^{y}\#\|^{z}\#\|^{s}\#\|^{t}\#\rightarrow$		$q_0\|^{x}\#\|^{y}\#\|^{z}\#\|^{s}\#\|^{t}\#\rightarrow$		$q_0\|^{x}\#\|^{y}\#\|^{z}\#\|^{s}\#\|^{t}\#\rightarrow$
1	$q_k\|^{x-t}\#\|^{z+s}\#$	16	$q_k\|^{y-s}\#\|^{x+z}\#$	31	$q_k\|^{x-z}\#\|^{y}\ ^{s}\#$	46	$q_k\|^{t-z}\#\|^{y}\ ^{s}\#$
2	$q_k\|^{x-}\#\|^{z+t}\#$	17	$q_k\|^{y}\ ^{s}\#\|^{x+t}\#$	32	$q_k\|^{x+z}\#\|^{y}\ ^{s}\#$	47	$q_k\|^{x+z}\#\|^{y}\ ^{s}\#$
3	$q_k\|^{x-}\#\|^{s+y}\#$	18	$q_k\|^{y}\ ^{s}\#\|^{z+t}\#$	33	$q_k\|^{z+t}\#\|^{y}\ ^{s}\#$	48	$q_k\|^{z+t}\#\|^{y}\ ^{s}\#$
4	$q_k\|^{x-}\#\|^{y+s}\#$	19	$q_k\|^{y}\ ^{t}\#\|^{x+z}\#$	34	$q_k\|^{x+z}\#\|^{y}\ ^{t}\#$	49	$q_k\|^{s+z}\#\|^{y}\ ^{t}\#$
5	$q_k\|^{x-}\#\|^{z+t}\#$	20	$q_k\|^{y}\ ^{t}\#\|^{x+s}\#$	35	$q_k\|^{x+s}\#\|^{y}\ ^{t}\#$	50	$q_k\|^{x+s}\#\|^{y}\ ^{t}\#$
6	$q_k\|^{x-}\#\|^{y+t}\#$	21	$q_k\|^{y}\ ^{t}\#\|^{z+s}\#$	36	$q_k\|^{z+s}\#\|^{y}\ ^{t}\#$	51	$q_k\|^{z+x}\#\|^{y}\ ^{t}\#$
7	$q_k\|^{x-}\#\|^{z+t}\#$	22	$q_k\|^{z}\ ^{t}\#\|^{x+s}\#$	37	$q_k\|^{s+y}\#\|^{z}\ ^{x}\#$	52	$q_k\|^{t+y}\#\|^{x}\ ^{s}\#$
8	$q_k\|^{x-}\#\|^{y+t}\#$	23	$q_k\|^{z}\ ^{x}\#\|^{y+t}\#$	38	$q_k\|^{x+s}\#\|^{z}\ ^{t}\#$	53	$q_k\|^{x+s}\#\|^{x}\ ^{t}\#$
9	$q_k\|^{x-}\#\|^{z+t}\#$	24	$q_k\|^{z}\ ^{x}\#\|^{y+s}\#$	39	$q_k\|^{y+t}\#\|^{z}\ ^{x}\#$	52	$q_k\|^{y+s}\#\|^{z}\ ^{t}\#$
10	$q_k\|^{x-t}\#\|^{y+s}\#$	25	$q_k\|^{z}\ ^{t}\#\|^{x+s}\#$	49	$q_k\|^{x+y}\#\|^{z}\ ^{t}\#$	55	$q_k\|^{x+y}\#\|^{z}\ ^{t}\#$
11	$q_k\|^{x-s}\#\|^{y+t}\#$	26	$q_k\|^{z}\ ^{t}\#\|^{x+s}\#$	41	$q_k\|^{x+s}\#\|^{z}\ ^{t}\#$	56	$q_k\|^{x+s}\#\|^{y}\ ^{t}\#$
12	$q_k\|^{x-t}\#\|^{y+s}\#$	27	$q_k\|^{z}\ ^{x}\#\|^{y+s}\#$	42	$q_k\|^{y+s}\#\|^{z}\ ^{t}\#$	57	$q_k\|^{y+s}\#\|^{z}\ ^{t}\#$
13	$q_k\|^{y-s}\#\|^{x+t}\#$	28	$q_k\|^{x}\ ^{t}\#\|^{y+s}\#$	43	$q_k\|^{x+y}\#\|^{z}\ ^{t}\#$	58	$q_k\|^{x+s}\#\|^{z}\ ^{t}\#$
14	$q_k\|^{y-t}\#\|^{x+s}\#$	29	$q_k\|^{s}\ ^{x}\#\|^{t+z}\#$	44	$q_k\|^{t+z}\#\|^{s}\ ^{x}\#$	59	$q_k\|^{x+z}\#\|^{y}\ ^{t}\#$
15	$q_k\|^{y-s}\#\|^{x+t}\#$	30	$q_k\|^{s}\ ^{y}\#\|^{t+z}\#$	45	$q_k\|^{x+z}\#\|^{s}\ ^{y}\#$	60	$q_k\|^{y+z}\#\|^{x}\ ^{t}\#$

Литература

1. Алферова Э.В. Теория алгоритмов. – М.: Статистика, 1973г.
2. Гуц А.К. Математическая логика и теория алгоритмов: Учебное пособие. – Омск: Изд-во Наследие, Диалог-Сибирь, 2003.
3. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженера. – 2-е изд.,перераб. и доп. – М.:Энергоатомиздат, 1988.
4. Математическая энциклопедия /Гл. ред. И. М. Виноградов. – М.: Советская энциклопедия. т.4. 1984.
5. Носов В.А. Основы теории алгоритмов и анализа их сложности. Курс лекций. –vnosov@intsys.msu.ru, 1992.
6. Першиков В.И., Савинков В.М. Толковый словарь по информатике.- М.: Финансы и статистика, 1991г.
7. Пономарев В.Ф. Модели вычислительных алгоритмов. Учебное пособие. – Калининград: Изд-во КГТУ, 1998г.
8. Словарь по кибернетике. /Под ред. В.С. Михалевича. – 2-е изд. – К.:

Гл. ред УСЭ им. М.П. Бажана, 1989г.

9. Структура данных и алгоритмы. Альфред В. Ахо, Джон Э. Хипкрофт, Джеффрид Д. Ульман. /пер. с англ. А.А. Минько/. –М.- СПб –Киев: издательский дом «Вильямс», 2003г.
10. Успенский В.А., Семенов А.Л. Теория алгоритмов: основные понятия и приложения. – М.: Наука, 1987г.

Оглавление