Глава 4. ТЕОРИЯ АЛГОРИТМОВ

Слово ***алгоритм*** (***алгорифм***) происходит от имени узбекского математика ***Аль-Хорезми,*** который в IX веке систематизировал правила арифметических операций. Полное имя:

Аль-Хорезми Абу Абдалла Мухаммед бен Муса аль - Маджуси.

«Китаб мухтасар аль-джебр ва-л-мукабала (Краткая книга восполнения и протовоставлнения)»

Аль-Хорезми

ПОНЯТИЕ АЛГОРИТМА

Под *алгоритмом* понимают точное предписание о выполнении в определенном порядке системы операций для решения всех задач некоторого данного типа.

Характеристика алгоритма:

- **алгоритм задается как инструкция конечных размеров;**
- **имеется вычислитель;**
- **имеется множество входных данных;**
- **имеется возможность запоминания;**
- **вычисления проводятся только по заданным инструкциям;**
- **в результате получается множество выходных данных.**

Алгоритм обладает свойствами *массовости, общедоступности и механистичностью.*

АЛФАВИТ, СЛОВА, АЛГОРИТМ В АЛФАВИТЕ

Алфавит – конечное множество различных символов (называемых **буквами**).

Примеры: {А, Б, В, Г, Д}; $\{\Theta, \Sigma, \varsigma, \Omega, \Psi\}$; $\{0, \Rightarrow, \&, 1\}$.

Слово в алфавите – конечная последовательность букв этого алфавита.

Примеры: $A = \{a, b, c, d\}$. $P = abb$, $Q = dda$, $PQ = abbdda$.

$$P\Lambda = \Lambda P = P.$$

Алгоритм в алфавите A перерабатывает слово (слова) из A в слово (слова) этого алфавита.

АЛГОРИТМ В АЛФАВИТЕ.

ВПОЛНЕ ЭКВИВАЛЕНТНЫЕ АЛГОРИТМЫ

Под *алгоритмом* $\mathcal{A}$ *в алфавите A* понимается алгоритм, входами и выходами которого являются слова в алфавите *A*.

$$\mathcal{A}\ (P) = Q$$

Слова в алфавите *A*

Алгоритм $\mathcal{A}$ применим к слову *P*, если преобразование *P* заканчивается через конечное число шагов: $Q = \mathcal{A}\ (P)$. Если процесс преобразования бесконечен, то алгоритм не применим к этому слову.

Два алгоритма $\mathcal{A}$ и $\mathcal{B}$ в одном и том же алфавите *C* наз-ся *вполне эквивалентными в алфавите D* ($D \subseteq C$), если для $\forall$ слова *P* в алфавите *D* оба алгоритма либо не применимы к *P*, либо применимы и их результаты совпадают:

$$\boxed{\forall P \text{ в } D:\ \ \mathcal{A}\ (P) \cong \mathcal{B}\ (P).}$$

НОРМАЛЬНЫЙ АЛГОРИТМ (АЛГОРИТМ А. А. МАРКОВА)

A - алфавит, не содержащий в качестве букв символов "$\bullet$" и "$\to$".

P и Q - слова в алфавите A. $P \to Q$ - *простая подстановка;*

$P \to \bullet Q$ - *заключительная подст-ка;*

$P \to (\bullet)\, Q$ - любая из $P \to Q$ или $P \to \bullet Q$.

$$B = \begin{cases} P_1 \to (\bullet) Q_1 \\ P_2 \to (\bullet) Q_2 \\ \dots\dots\dots\dots \\ P_n \to (\bullet) Q_n \end{cases}$$

$n \geq 1$, P_i и Q_i, $(1 \leq i \leq n)$ - слова в A.

Пусть $A = \{ a, b, c \}$ и

$$B = \begin{cases} a \to a & (1) \\ cc \to (\bullet) c & (2) \\ b \to c & (3) \end{cases}$$

Если $R_o = bb$,

cb – по (3)

cc - по (3)

c - по (2) Следовательно: $B(R_o) = c$.

ПРИМЕР НОРМАЛЬНОГО АЛГОРИТМА

Пусть $A = \{\, a, b, c \,\}$ и

$$B = \begin{cases} \beta a \rightarrow a\beta & (1) \\ \beta b \rightarrow b\beta & (2) \\ \beta c \rightarrow c\beta & (3) \\ \beta \rightarrow \bullet a & (4) \\ \Lambda \rightarrow \beta & (5) \end{cases}$$

где $\beta \notin A$.

Алгоритм:

$$B^* = \begin{cases} \beta x \rightarrow x\beta \quad (x \in A) & (1^*) \\ \beta \rightarrow (\bullet)a & (2^*) \\ \Lambda \rightarrow \beta & (3^*) \end{cases}$$

Здесь $\beta x \rightarrow x\beta$ для обозначения подстановок *(1) - (3)* из *B*.

ПРИМЕР НОРМАЛЬНОГО АЛГОРИТМА

Пусть $M = \{ 1, * \}$. Число 0 обозначим словом $\overline{0} = 1$,

число 1 обозначим словом $\overline{1} = 11$,

....

число n обозначим словом $\overline{n} = \underbrace{11 \ldots 1}_{n+1\,единиц}$.

вектор $(n_1, n_2, \ldots, n_k) \leftrightarrow \overline{n}_1 * \overline{n}_2 * \ldots * \overline{n}_k \leftrightarrow \overline{(n_1, n_2, \ldots, n_k)}$. Напр.

$\overline{(1, 2, 3)} = 11*111*1111.$ $\quad A_0 = \begin{cases} * \to * \\ \alpha 11 \to \alpha 1 \\ \alpha 1 \to \bullet 1 \\ \Lambda \to \alpha \end{cases}$

применим только к тем словам в алфавите M, которые суть цифры, переводит $\overline{n}$ в $\overline{0}$ и A_0 не применим к пустому слову.

Нормальный алгоритм: $A_1 = \begin{cases} * \to * \\ \alpha 1 \to \bullet 11 \\ \Lambda \to \alpha \end{cases}$

преобразует $\forall\ \overline{n}$ в $\overline{n+1}$ и A_1 не применим к пустому слову.

ФУНКЦИИ ЧАСТИЧНО - ВЫЧИСЛИМЫЕ И ВЫЧИСЛИМЫЕ ПО МАРКОВУ

Частично определенная функция $f(x_1,x_2,\dots,x_n)$ называется *частично вычислимой по Маркову*, если существует нормальный алгоритм B над $M = \{ 1, * \}$:

$$B((\overline{k_1,k_2,\dots,k_n})) = \overline{f(k_1,k_2,\dots,k_n)}$$

тогда и только тогда, когда хотя бы одна из частей этого равенства определена.

n **- аргументная функция *f вычислима по Маркову*, когда $\exists$ норм. алгоритм, позволяющий вычислить значение $f(x_1, x_2, \dots, x_n)$ для $\forall$ совокупностей значений $x_1, x_2, \dots, x_n$.**

$f(x) = 0$, $\varphi(x) = x + 1$, $\forall x\ (x \geq 0)$ - функции вычислимыми по Маркову.

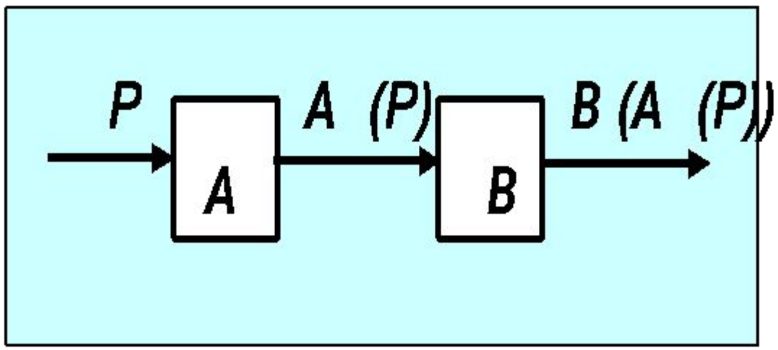
Замыкание алгоритма:

$$A = \begin{cases} P_1 \to (\bullet)Q_1 \\ P_2 \to (\bullet)Q_2 \\ \dots\dots\dots\dots \\ P_n \to (\bullet)Q_n \end{cases} \qquad \text{его замыкание:} \qquad A^{\bullet} = \begin{cases} P_1 \to (\bullet)Q_1 \\ P_2 \to (\bullet)Q_2 \\ \dots\dots\dots\dots \\ P_n \to (\bullet)Q_n \\ \Lambda \to \bullet\Lambda \end{cases}$$

Алгоритмы A и $A^{\bullet}$ вполне эквивалентны.

КОМПОЗИЦИЯ АЛГОРИТМОВ

Композицией **алгоритмов A и B в алфавите A называют алгоритм C такой, что $\forall P$ в $A: C(P) \cong B(A(P))$.**



Композиция алгоритмов A и B обозначается как: $C = B \circ A$.

$$A_n \circ A_{n-1} \circ \ldots \circ A_1 =$$

$$= A_n \circ (A_{n-1} \circ (\ldots \circ (A_3 \circ (A_2 \circ A_1))\ldots)).$$

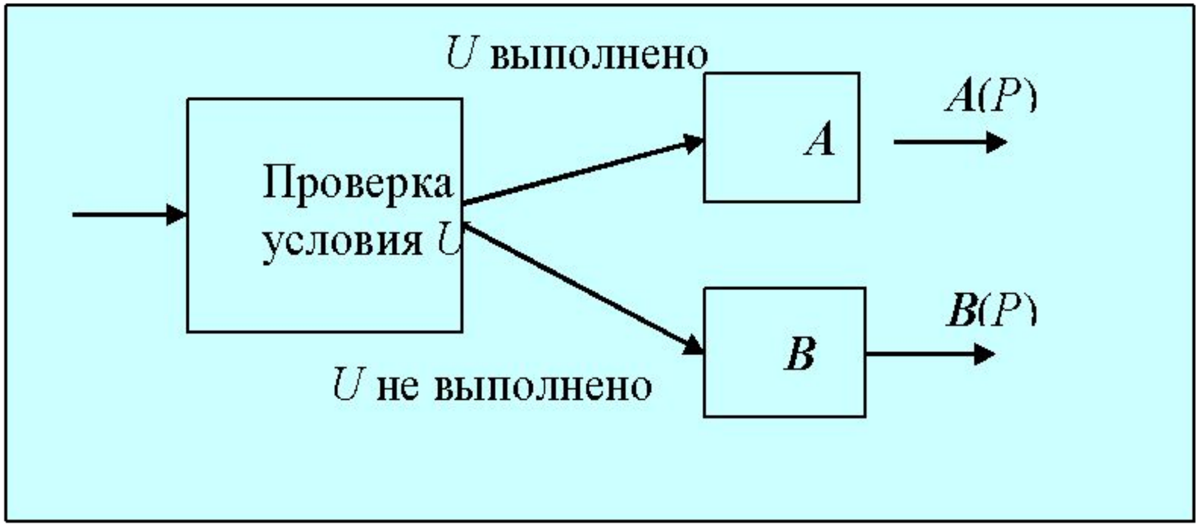
Теорема. Композиция нормальных алгоритмов $A_1, A_2, \ldots, A_n$ в алфавите A есть снова нормальный алгоритм (над алфавитом A).

КОМПОЗИЦИЯ АЛГОРИТМОВ

$$C = \begin{cases} a\alpha \to \alpha a & (a \in A) & (1) \\ \alpha a \to \alpha \bar{a} & (a \in A) & (2) \\ \bar{a}b \to \bar{a}\bar{b} & (a, b \in A) & (3) \\ \bar{a}\beta \to \beta \bar{a} & (a \in A) & (4) \\ \beta\bar{a} \to \beta a & (a \in A) & (5) \\ a\bar{b} \to ab & (a, b \in A) & (6) \\ \alpha\beta \to \bullet\Lambda & & (7) \\ \bar{B}^{\beta} & & (8) \\ A^{\alpha} & & (9) \end{cases}$$

РАЗВЕТВЛЕНИЕ АЛГОРИТМОВ

Пусть заданы алгоритмы *A* и *B* в алфавите *M* и некоторое условие *U*.

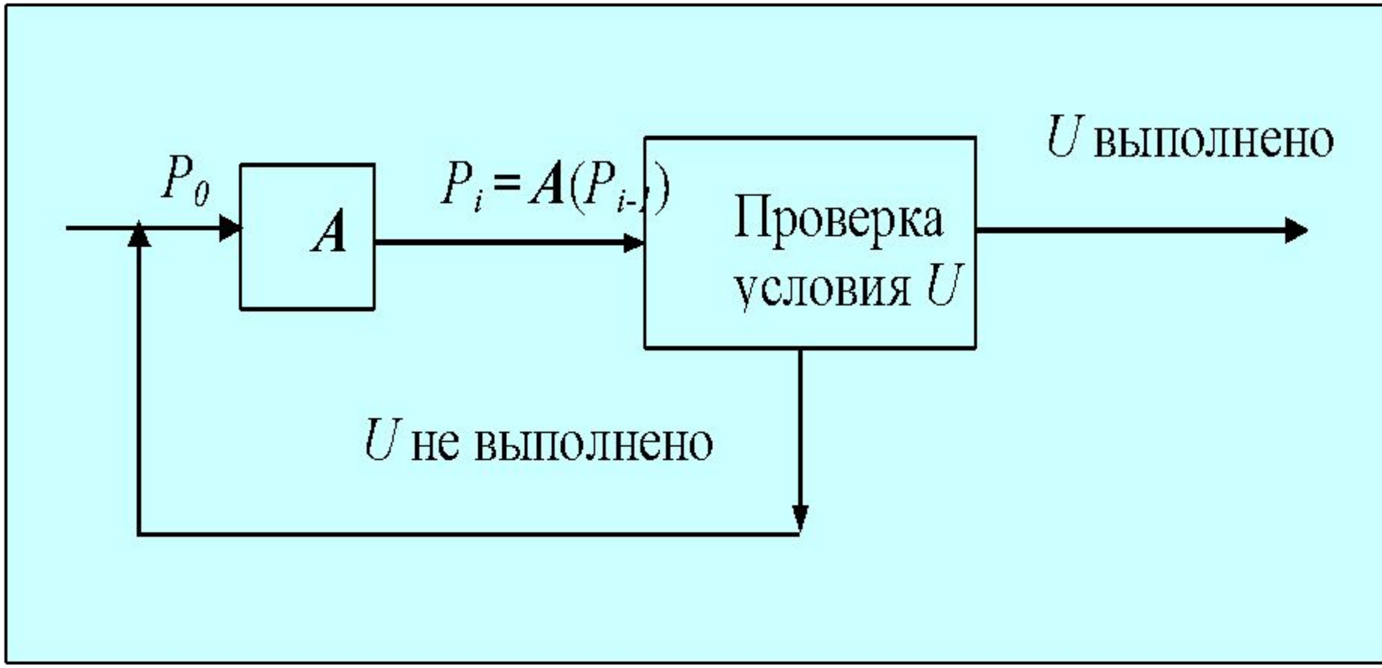


Условие *U* для слова *P* выполнено, если *C (P)* = Λ,
условие *U* для слова *P* не выполнено, если *C (P)* ≠ Λ.

$$\forall P \text{ в } M: \; R(P) \cong \begin{cases} A(P), \text{ если } C(P) = \Lambda; \\ B(P), \text{ если } C(P) \neq \Lambda. \end{cases}$$

Теорема. Разветвление нормальных алгоритмов, управляемое нормальным алгоритмом, является нормальным алгоритмом.

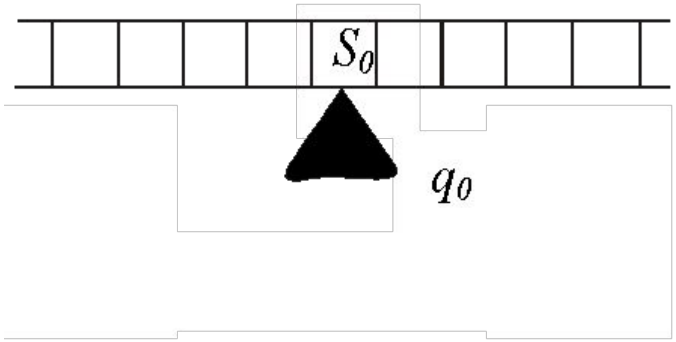
ПОВТОРЕНИЕ АЛГОРИТМОВ



Повторение алгоритмов

Теорема. Повторение нормального алгоритма, управляемое нормальным алгоритмом, есть нормальный алгоритм.

МАШИНА ТЬЮРИНГА



$A = \{ S_0, S_1, ..., S_n \}$ – *(внешний) алфавит* машины

$\{ q_0, q_1, ..., q_m \}$ - множество *внутренних состояний*.

Действия машины:

1) головка стирает символ S_i и записывает там же символ S_k;

2) головка перемещается в соседний слева квадрат;

3) головка перемещается в соседний справа квадрат;

4) машина останавливается.

В случаях 1) - 3) машина переходит во внутреннее состояние q_r и готова к действию в следующий момент времени $t + 1$.

1) $q_j\, S_i\, S_k\, q_r$;

2) $q_j\, S_i\, L\, q_r$;

3) $q_j\, S_i\, R\, q_r$.

ЗАДАНИЕ МАШИНЫ ТЬЮРИНГА

Машина Тьюринга *T* заданна, если задано непустое конечное множество команд, удовлетворяющих условиям:

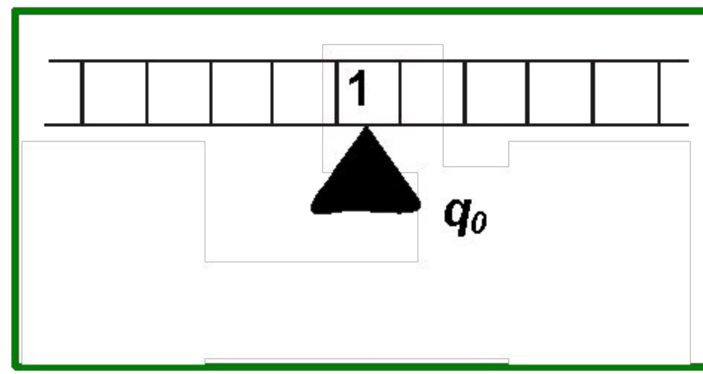
1) никакие две команды не имеют совпадающие первые два символа;

2) среди команд есть хотя бы одна команда, начинающаяся с q_0.

Машина останавливается, если она находится в состоянии q_j, обозревает символ S_k , а среди команд машины нет команды, начинающейся с $q_j S_k$.

1. *M T* :

$$q_0\, 1\, R\, q_1$$
$$q_1\, S_0\, 1\, q_0$$



2. *M T:*

$$q_0\, a\, R\, q_0$$
$$q_0\, b\, R\, q_0$$
$$q_0\, S_0\, a\, q_1$$

приписывает к ∀ слову *P* алфавита { *a*, *b* } справа букву *a* и останавливается.

АЛГОРИТМ ТЬЮРИНГА. ВЫЧИСЛИМОСТЬ ПО ТЬЮРИНГУ

Частично определённая арифметическая функция $f(x_1, x_2, ..., x_n)$ **наз-ся *частично вычислимой по Тьюрингу* если ∃ машина Тьюринга *T* с алфавитом *M*, включающим 1 и *, такая, что для ∀ натуральных** $k_1, k_2, ..., k_n$ **и некоторых** r **и** m **имеем:**

$$A_{T,M}(\overline{k_1, k_2, ..., k_n}) = S_0^r\, \overline{f(k_1, k_2, ..., k_n)}\, S_0^m \quad (S_0^i = \underbrace{S_0 S_0 ... S_0}_{i\ \text{раз}}),$$

т. и т. т., когда определена хотя бы одна из частей этого равенства.

Арифметическая функция $f(x_1, x_2, ..., x_n)$ **наз-ся *вычислимой по Тьюрингу*, если ∃ машина Тьюринга *T* с алфавитом *M*, включающим 1 и *, такая, что для ∀ натуральных** $k_1, k_2, ..., k_n$ **найдутся ∃** r **и** m**:**

$$A_{T,M}(\overline{k_1, k_2, ..., k_n}) = S_0^r\, \overline{f(k_1, k_2, ..., k_n)}\, S_0^m.$$

СВЯЗЬ МЕЖДУ МТ И НОРМАЛЬНЫМИ АЛГОРИТМАМИ

Теорема. Пусть T - машина Тьюринга с алф. M. Тогда $\exists$ нормальный алгоритм B над A, вполне эквивалентный относительно M алгоритму Тьюринга $A_{T,M}$.

Следствие. Всякая частично вычислимая (вычислимая) по Тьюрингу функция является частично вычислимой (вычислимой) по Маркову.

Пусть $f(x_1, x_2, \ldots, x_n)$ вычислима по Тьюрингу: $A_{T,M}(\overline{k_1, k_2, \ldots, k_n}) = R_1 \overline{f(k_1, k_2, \ldots, k_n)} R_2$. Тогда $\exists$ нормальный алгоритм B: $B \cong A_{T,A}$:

$$A_{T,M}((\overline{k_1, k_2, \ldots, k_n})) \cong B((\overline{k_1, k_2, \ldots, k_n})) \cong R_1 \overline{f(k_1, k_2, \ldots, k_n)} R_2.$$

$$B_1 = \begin{cases} \alpha S_0 \to \alpha \\ \alpha 1 \to \bullet 1 \\ \alpha * \to \bullet * \\ \alpha \to \bullet \Lambda \\ \Lambda \to \alpha \end{cases} \qquad B_2 = \begin{cases} \alpha * \to * \alpha \\ \alpha 1 \to 1\alpha \\ \alpha S_0 \to \alpha \\ \alpha \to \bullet \Lambda \\ \Lambda \to \alpha \end{cases} \qquad C = B_2 \bullet B_1 \bullet B. \qquad \text{Тогда:}$$

$$B\left((\overline{k_1, k_2, \ldots, k_n})\right) \cong A_{T,M}\left((\overline{k_1, k_2, \ldots, k_n})\right) \cong R_1 \overline{f(k_1, k_2, \ldots, k_n)} R_2,$$

$$B_1\left(R_1 \overline{f(k_1, k_2, \ldots, k_n)} R_2\right) = \overline{f(k_1, k_2, \ldots, k_n)} R_2;$$

$$B_2\left(\overline{f(k_1, k_2, \ldots, k_n)} R_2\right) = \overline{f(k_1, k_2, \ldots, k_n)}.$$

ОСНОВНАЯ ГИПОТЕЗА ТЕОРИИ АЛГОРИТМОВ

(ПРИНЦИП НОРМАЛИЗАЦИИ ИЛИ ТЕЗИС ЧЕРЧА)

Для всякого алгоритма *B* в алфавите *M* существует вполне эквивалентный ему нормальный алгоритм *C* над *M*, т.е.

$$\forall P \text{ в } M: \ B(P) \cong C(P).$$

Иначе эта гипотеза формулируется так:

Всякий алгоритм может быть задан посредством некоторой машины Тьюринга и реализован в этой машине.

АЛГОРИТМИЧЕСКИ РАЗРЕШИМЫЕ ПРОБЛЕМЫ:

-сложение двух и более заданных чисел;

-решение в радикалах уравнений от одной переменной не выше четвертой степени;

-решение систем линейных уравнений с *n* неизвестными;

-и т. д.

АЛГОРИТМИЧЕСКИ НЕРАЗРЕШИМЫЕ ПРОБЛЕМЫ:

1. Проблема *диофантовых корней* (10-ая проблема Гильберта)

$$P_n(x_1, x_2, \ldots, x_m) = 0$$

В 1970 году советским математиком Ю.В. Матиясевичем было доказано, что эта проблема алгоритмически неразрешима.

2. *Проблема распознавания применимости.* *A (P)* -- ?

> Теорема. Не существует нормального алгоритма *B*, который позволил бы выяснить, применим или нет произвольный нормальный алгоритм *A* к произвольному слову *P*.

АЛГОРИТМИЧЕСКИ НЕРАЗРЕШИМЫЕ ПРОБЛЕМЫ:

3. *Проблема эквивалентности слов*: для любых двух слов в данном алфавите требуется указать, эквивалентны они или нет - проблема эквивалентности слов. Марков и Пост доказали, что данная проблема алгоритмически неразрешима.

4. *Проблема неразрешимости логики предикатов*. Черчем доказано, что не существует алгоритма, который для любой формулы логики предикатов устанавливает, логически общезначима она или нет.

5. *Проблема остановки*. Тьюрингом доказано, что не существует алгоритма, позволяющего выяснить: остановится или нет произвольная программа для произвольного заданного входа. Для теоретического программирования означает, что не существует общего метода проверки программ на наличие в них бесконечных циклов.

6. Не существует алгоритма, позволяющего установить, вычисляет ли некоторая программа постоянную нулевую функцию $Z(x) = 0$. Тогда проблема вычисляют ли две произвольные программы одну и ту же одноаргументную функцию, тоже алгоритмически неразрешима.

РЕКУРСИВНЫЕ ФУНКЦИИ

Исходные функции:

1) ***нуль функция***: $Z(x) = 0$ при $\forall x\ (x \geq 0)$,
2) ***функция прибавления единицы***: $N(x) = x + 1$ при $\forall x\ (x \geq 0)$,
3) ***проектирующая функция*** $J_i^n(x_1, x_2, \ldots, x_n) = x_i$ при $\forall x_1, x_2, \ldots, x_n \geq 0$.

Правила для получения новых функций:

Подстановка: $f(x_1, x_2, \ldots, x_n) =$

$$g(h_1(x_1, x_2, \ldots, x_n), \ldots, h_m(x_1, x_2, \ldots, x_n)).$$

Рекурсия:

$$\text{а)} \begin{cases} f(x_1, x_2, \ldots, x_n, 0) = g(x_1, x_2, \ldots, x_n), \\ f(x_1, x_2, \ldots, x_n, y + 1) = h(x_1, x_2, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y)), \end{cases}$$

для $n = 0$:

$$\text{б)} \begin{cases} f(0) = k, \text{ где } k \text{ - Const}, \\ f(y + 1) = h(y, f(y)). \end{cases}$$

μ - оператор: пусть $g(x_1, x_2, \ldots, x_n, y)$ такова, что для $\forall x_1, x_2, \ldots, x_n\ \exists\ y$: $g(x_1, x_2, \ldots, x_n, y) = 0$. Обозначим через $\mu y\ (g(x_1, x_2, \ldots, x_n, y) = 0)$ наименьшее значение y, при котором $g(x_1, x_2, \ldots, x_n, y) = 0$. Полагаем:

$$f(x_1, x_2, \ldots, x_n) = \mu\ y\ (g(x_1, x_2, \ldots, x_n, y) = 0).$$

РЕКУРСИВНЫЕ ФУНКЦИИ

Функция называется *примитивно рекурсивной*, если она может быть получена из исходных функций 1), 2) и 3) с помощью конечного числа подстановок и рекурсий.

Функция f называется *общерекурсивной*, если она может быть получена из исходных функций 1), 2) и 3) с помощью конечного числа подстановок, рекурсий и μ - оператора. Общерекурсивные функции иногда называют рекурсивными функциями.

Частичная функция φ называется *частично рекурсивной*, если она может быть получена из исходных функций 1), 2) и 3) с помощью конечного числа подстановок, рекурсий и μ' -оператора, где μ' - оператор определяется как и μ -оператор, но при некоторых значениях $x_1, x_2, \ldots, x_n$ может и не существовать y такого, что $g(x_1, x_2, \ldots, x_n, y) = 0$.

Теорема. Следующие функции являются примитивно рекурсивными:

1) $x + y$;

2) $x \bullet y$;

3) x^{y};

4) $$\delta(x) = \begin{cases} x - 1, \text{ если } x > 0, \\ 0, \quad \text{если } x = 0; \end{cases}$$

5) $$x \dot{-} y = \begin{cases} x - y, & x \geq y, \\ 0, & x < y; \end{cases}$$

6) $$|x - y| = \begin{cases} x - y, & x \geq y \\ y - x, & x < y; \end{cases}$$

7) $$Sg(x) = \begin{cases} 0, & x = 0, \\ 1, & x \neq 0; \end{cases}$$

8) $$sg^*(x) = \begin{cases} 1, & x = 0, \\ 0, & x \neq 0; \end{cases}$$

9) $rm(x, y)$ = остатку от деления y на x;

10) $qt(x, y)$ = частному от деления y на x;

11) $x!$;

12) $min(x, y)$;

13) $min(x_1, x_2, \ldots, x_n)$;

14) $max(x, y)$;

15) $max(x_1, x_2, \ldots, x_n)$.

ТЕОРИЯ АЛГОРИТМОВ

Известные советские математики Успенский В. А. и Семенов А. Л. в обзоре основных достижений теории алгоритмов пишут:

"***Алгоритмические концепции играют в процессе обучения и воспитания современного человека фундаментальную роль, сравнимую лишь с ролью письменности***".