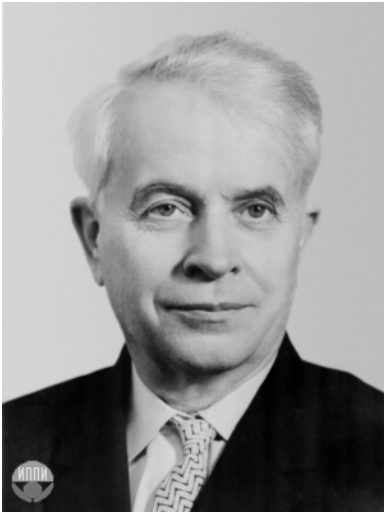
6.6. НОРМАЛЬНЫЕ АЛГОРИТМЫ МАРКОВА



Марков
Андрей Андреевич
(9 [22] сентября 1903 г. –
11 октября 1979 г.)

Советский математик, основоположник советской школы конструктивной математики.

Основные труды – по теории динамических систем, топологии, топологической алгебре, теории алгоритмов и конструктивной математике.

Доказал неразрешимость проблемы равенства в ассоциативных системах (1947), проблемы гомеоморфии в топологии (1958), создал школу конструктивной математики и логики в СССР.

Является автором принципа Маркова, нормального алгоритма Маркова, теоремы Маркова-Какутани и теоремы Риса-Маркова-Какутани.

Теория нормальных алгоритмов (или алгорифмов, как назвал их создатель теории) был разработана советским математиком А.А. Марковым в конце 1940-х – начале 1950-х гг. Эти алгоритмы представляют собой некоторые правила по переработке слов в каком-либо алфавите, так что исходные данных и искомый результат для алгоритмов являются словами в некотором алфавите.

Будем называть **алфавитом** всякое непустое конечное множество символов, а сами символы алфавита будем называть *буквами*.

Словом в алфавите A называется всякая последовательность букв алфавита A.

Пустая последовательность букв называется **пустым словом** и обозначается через Λ (лямбда).

Слова будем обозначать заглавными латинскими буквами P, Q, R (или этими же буквами с индексами).

Одно слово может быть составной частью другого слова. Тогда первое называется **подсловом** второго или вхождением во второе.

Например, если A – алфавит русских букв, то можно рассмотреть такие слова:

$$P_1 = \text{параграф},$$
$$P_2 = \text{граф},$$
$$P_3 = \text{ра}.$$

Слово P_2 является подсловом слова P_1:

параграф

Слово P_3 является подсловом слов P_1 и P_2, причем в слово P_1 оно входит дважды.

парагра ф

гра ф

Марковской подстановкой называется операция над словами, которая задается с помощью упорядоченной пары слов (P, Q) и заключается в замене на слово Q первого вхождения слова P (если таковое имеется) в слово R, при этом остальные части слова R остаются без изменений. Полученное слово называется **результатом применения марковской подстановки (P, Q) к слову R**.

Если же первого вхождения P в слово R нет (и, следовательно, вообще нет ни одного вхождения P в R), то считается, что **марковская подстановка (P, Q) неприменима к слову R**.

Если P обозначает слово $S_{i1}\ldots S_{ik}$, а Q – слово $S_{r1}\ldots S_{rm}$, то пусть PQ обозначает **соединение** $S_{i1}\ldots S_{ik}S_{r1}\ldots S_{rm}$ этих двух слов.

В частности: $P\Lambda=\Lambda P=P$; $(P_1P_2)P_3=P_1(P_2P_3)$.

Пример

Исходное слово ***R***	**Марковская подстановка** **(*P*, *Q*)**	**Результат подстановки**
138578926	(85789, 00)	130026
тараарам	(ара, Λ)	Трам
шрам	(ра, ар)	Шарм
функция	(Λ, ζ-)	ζ-функция
логика	(ика, Λ)	Лог
книга	(Λ, Λ)	книга
поляна	(пор, т)	[*неприменима*]

Алфавит A называется **расширением алфавита *B***, если $B \subseteq A$ (т.е. B – подмножество A).

Алгоритмом в алфавите *A* называется эффективно вычислимая функция, областью определения которой служит какое-нибудь подмножество множества всех слов в алфавите A.

Пусть P есть слово в алфавите A; говорят, что **алгоритм α применим к слову *P***, если P содержится в области определения α.

Если алфавит B является расширением алфавита A, то всякий алгоритм в алфавите B называется **алгоритмом над алфавитом *A***.

Большинство известных алгоритмов можно разбить на некоторые простейшие шаги. Каждый шаг алгоритма представляет собой реализацию какой-то определенной операции. В качестве элементарной операции, на базе которой будут строиться алгоритмы, рассмотрим операцию подстановки одного слова вместо другого.

Если P и Q – слова в алфавите A, то выражение

$$P \to Q \text{ и } P \to\bullet\ Q$$

будем называть **формулами подстановки в алфавите A** (причем символы $\to$ и $\bullet$ не являются буквами алфавита A).

Формула $P \to Q$ называется **простой подстановкой**; $P \to\bullet\ Q$ – **заключительной**.

Конечный список формул подстановки в алфавите A

$$\begin{cases} P_1 \to (\bullet) Q_1, \\ P_2 \to (\bullet) Q_2, \\ \ldots \\ P_r \to (\bullet) Q_r, \end{cases}$$

где $\to(\bullet)$ обозначает одну из операций $\to$ или $\to\bullet$, называется **схемой алгоритма** и порождает следующий алгоритм в алфавите A.

Слово *T* входит в слово *Q*, если существуют такие (возможно пустые) слова U, V, при которых

$$Q = UTV.$$

Примечание. В некоторых учебниках заключительную подстановку обозначают такой стрелкой:

$$\mapsto$$

Пусть P – некоторое слово в алфавите A.

Будем использовать следующие обозначения:

1. Ни одно из слов может не входить в слово P. Этот факт будем обозначать следующим образом:

$$\alpha: P\ ].$$

2. Пусть среди слов $P_1,\ldots P_r$ существуют слова, которые входят в P, m – наименьшее целое число (такое, что $1 \le m \le r$) и P_m входит в P. R – слово, которое получается, если самое левое вхождение слова P_m в слово P заменить Q_m. Тот факт, что P и R находятся в данном отношении, коротко записывается так:

а) $\alpha: P \vdash R,$

если формула подстановки $P_m \to (\bullet) Q_m$ – простая (тогда мы будем говорить, что алгоритм α просто переводит слово P в слово R), или в виде

б) $\alpha: P \vdash\!\!\bullet\ R$,

если формула подстановки $P_m \to (\bullet) Q_m$ – заключительная (тогда мы будем говорить, что алгоритм α заключительно переводит слово P в слово R).

Пусть

$$\alpha: P \models R$$

означает, что существует такая последовательность $R_0, R_1, \ldots R_k$ слов в алфавите A, что

$$P = R_0,\ R = R_k,\ \alpha: R_j \vdash R_{j+1} \text{ для } j = 0, 1, \ldots, k-1.$$

$$\alpha: P \models\!\!\bullet\ R$$

означает, что существует такая последовательность $R_0, R_1, \ldots R_k$ слов в алфавите A, что

$$P = R_0,\ R = P_k,\ \alpha: R_j \vdash R_{j+1} \text{ для } j = 0, 1, \ldots, k-2,\ \boldsymbol{R_{k-1} \vdash\!\!\bullet\ R_k}.$$

Будем полагать, что алгоритм α применим к слову P и дает в результате слово R ($\alpha(P) = R$) тогда и только тогда, когда

либо $\alpha: P \mid= \bullet R$,

либо $\alpha: P \mid= R$ и $\alpha: \mathbf{R}\ ]$.

Алгоритм, определенный таким образом, называется **нормальным алгоритмом Маркова в алфавите** A.

Работа алгоритма может быть описана следующим образом.

Пусть дано слово P в алфавите A.

Находим **первую** в схеме алгоритма α формулу подстановки $P_m \to (\bullet) Q_m$ такую, что P_m входит в P. Совершаем подстановку слова Q_m вместо самого левого вхождения слова P_m в слово P. Пусть R_1 – результат такой подстановки. Если $P_m \to (\bullet) Q_m$ – заключительная формула подстановки, то работа алгоритма заканчивается и его значением является R_1. Если формула подстановки $P_m \to (\bullet) Q_m$ простая, то применим к R_1 тот же поиск, который был применен к P, и т.д.

Если в конце концов будет получено такое слово R_i, что α: R_i], т.е. ни одно из слов $P_1, \ldots P_r$ не входит в P_i, то работа алгоритма заканчивается и R_i будет его значением.

При этом возможно, что описанный процесс никогда не закончится. В таком случае мы говорим, что алгоритм α неприменим к слову P.

Пример 6.19. Пусть A есть алфавит $\{a, b, c\}$.

Рассмотрим схему

$$\begin{cases} a \to b, \\ b \to \bullet\Lambda, \\ c \to c. \end{cases}$$

Пусть слово

$$\boldsymbol{P = abc.}$$

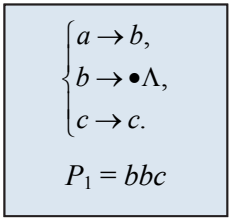
1. Рассмотрим 1-ю формулу алгоритма: $\boldsymbol{a \to b}$. Ищем первое вхождение подслова $\boldsymbol{a}$ в слово P**.** Такое подслово в слово P входит:

$$\boldsymbol{P = abc}$$

Выполняем подстановку $\boldsymbol{a \to b}$. В результате получаем слово

$$\boldsymbol{P_1 = bbc}$$

2. Текущее слово: $P_1 = bbc$. Проверяем возможность применения 1-й формулу алгоритма $\boldsymbol{a \to b}$ к слову P_1. Подслово $\boldsymbol{a}$ в слово P_1 не входит. Следовательно, данная команда алгоритма не применима к слову P_1.



3. Текущее слово: $P_1 = bbc$. Проверяем возможность применения 2-й формулы алгоритма $\boldsymbol{b \to \bullet\Lambda}$ к слову P_1. Ищем первое вхождение подслова $\boldsymbol{b}$ в слово P_1

$$\boldsymbol{P_1 = bbc}$$

Выполняем подстановку $\boldsymbol{b \to \bullet\Lambda}$. В результате получаем слово

$$\boldsymbol{P_2 = bc}$$

Так как подстановка $\boldsymbol{b \to \bullet\Lambda}$ была заключительной, то результатом работы алгоритма (словом R) будет слово P_2:

$$\boldsymbol{R = bc.}$$

Если бы схему алгоритма

$$\begin{cases} a \to b, \\ b \to \bullet\Lambda, \\ c \to c. \end{cases}$$

мы бы попробовали применить к слову

$$\boldsymbol{P = cc},$$

то получилась бы бесконечная последовательность слов:

$$cc,$$

$$cc,$$

$$cc,$$

$$\ldots$$

Данную задачу можно реализовать с помощью машины Тьюринга: *Пусть задан алфавит A = {a, b, c}. В непустом слове P необходимо сначала заменить все вхождения символа a на b, затем первое вхождение символа b удалить. В начальный момент обозревается самый левый символ слова P.*

$$\begin{cases} a \to b, \\ b \to \bullet\Lambda, \\ c \to c. \end{cases}$$

$$P_1 = bbc$$

Состояния множества *Q*	**Символы алфавита**			
	a	***b***	***c***	**λ**
q_1	q_1bR	q_1bR	q_1cR	$q_2\lambda L$
q_2	–	q_2bL	q_2cL	$q_3\lambda$R
q_3	–	$q_0\lambda R$	$q_4\lambda R$	–
q_4	–	q_0cR	q_4cR	q_0cE

q_1 – проход слева направо по слову и замена всех *a* на *b*; q_2 – проход по слову справа налево; q_3 – анализ первого символа слова (для слов с началом на букву *c* начало циклической перестановки букв *c* слове (до первой буквы *b*) с использованием запоминания символа с помощью состояний; для слов начинающихся с символа *b*, удаление этого символа и остановка); q_4 – «выгрузка из буфера» символа *c*.

В теории нормальных алгоритмов Маркова рассматриваются также операции композиции двух алгоритмов, проекции слова на алфавит и т.д.

Пусть α и β – алгоритмы, а P – слово. Обозначим через $\alpha(P) \sim \beta(P)$ факт того, что либо оба алгоритма α и β неприменимы к слову P, либо оба они к нему применимы и при этом $\alpha(P) = \beta(P)$. Назовем ***два алгоритма α и β над алфавитом A вполне эквивалентными относительно A***, если $\alpha(P) \sim \beta(P)$ для любого слова P в алфавите A.

Как и машины Тьюринга, нормальные алгоритмы не производят собственно вычислений: они лишь производят преобразования слов, заменяя в них буквы другими по предписанным правилам. В свою очередь, мы предписываем им такие правила, результаты которых мы можем интерпретировать как вычисления. Рассмотрим два примера.

Пример 6.20. В алфавите $A = \{1\}$ схема

$$\Lambda \rightarrow\bullet\ 1$$

определяет алгоритм, который к каждому слову в алфавите $A = \{1\}$ (все такие слова суть следующие: Λ, 1, 11, 111, 1111 и т.д.) предписывает слева 1.

Следовательно, алгоритм реализует (вычисляет) функцию $f(x) = x + 1$.

Для сравнения:

система команд машины Тьюринга, реализующей такую же операцию (в начальный момент обозревается самая левая единица числа):

Состояния множества Q	**Символы алфавита**	
	1	**λ**
q_1	$q_1 1 L$	$q_0 1 E$

Пример 6.21. Дана функция

$$\varphi_3(11...1) = \begin{cases} 1, \text{если } n \text{ делится на } 3, \\ 0, \text{если } n \text{ не делится на } 3. \end{cases}$$

где n – число единиц в слове 11…1. Рассмотрим нормальный алгоритм в алфавите $A = \{1\}$ со следующей схемой:

$$\begin{cases} 111 \to \Lambda, \\ 11 \to \bullet\, \Lambda, \\ 1 \to \bullet\, \Lambda, \\ \Lambda \to \bullet\, 1. \end{cases}$$

Этот алгоритм работает по такому принципу: пока число букв 1 в слове не меньше 3, алгоритм последовательно стирает по три буквы. Если число букв меньше 3, но больше 0, то оставшиеся буквы 1 или 11 стираются; если слово пусто, оно переводится в слово 1. Работа алгоритма на этом завершается.

Например:

$$\begin{cases} 111 \to \Lambda, \\ 11 \to \bullet\, \Lambda, \\ 1 \to \bullet\, \Lambda, \\ \Lambda \to \bullet\, 1. \end{cases}$$

$$1111111 \to 1111 \to 1 \to \Lambda.$$

$$111111111 \to 111111 \to 111 \to \Lambda \to 1.$$

Таким образом, рассмотренный алгоритм реализует (или вычисляет) функцию:

$$\varphi_3(11\ldots1) = \begin{cases} 1, \text{если } n \text{ делится на } 3, \\ 0, \text{если } n \text{ не делится на } 3. \end{cases}$$

где n – число единиц в слове 11…1.

Рассмотрим реализацию данного алгоритма с помощью машины Тьюринга (в начальный момент обозревается самая левая единица числа, в качестве входного числа 0 будем рассматривать пустое слово):

Состояния множества Q	**Символы алфавита**	
	1	**λ**
q_1	$q_2 \lambda L$	$q_0 1 E$
q_2	$q_3 \lambda L$	$q_0 0 E$
q_3	$q_1 \lambda L$	$q_0 0 E$

Для подсчета цифр 1 используется поочередная смена состояний q_1, q_2, q_3.

Сформулирует теперь точное определение такой вычислимости функции.

Функция *f*, заданная на некотором множестве слов алфавита A, называется **нормально вычислимой**, если найдется такое расширение B данного алфавита ($A \subseteq B$) и такой нормальный алгоритм α в B, при которых каждое слово V (в алфавите A) из области определения функции f этот алгоритм перерабатывает в слово $f(V)$.

Таким образом, нормальные алгоритмы, рассмотренные в примерах 6.20 и 6.21, нормально вычислимы. Причем соответствующие нормальные алгоритмы удалось построить в том же самом алфавите A, т.е. расширение алфавита не потребовалось ($B = A$).

Рассмотрим пример построения нормального алгоритма в расширенном алфавите.

Пример 6.22. Построить нормальный алгоритм для вычисления функции $f(x) = x + 1$ не в унарной системе (как уже было сделано), а в десятичной. В качестве алфавита возьмем перечень арабских цифр

$$A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\},$$

а нормальный алгоритм будем строить в его расширении

$$B = A \cup \{a, b\}.$$

Схема этого нормального алгоритма приведена ниже.

$0b \to \cdot 1$
$1b \to \cdot 2$
$2b \to \cdot 3$
$3b \to \cdot 4$
$4b \to \cdot 5$
$5b \to \cdot 6$
$6b \to \cdot 7$
$7b \to \cdot 8$
$8b \to \cdot 9$
$9b \to b0$
$b \to \cdot 1$
$a0 \to 0a$
$a1 \to 1a$
$a2 \to 2a$
$a3 \to 3a$
$a4 \to 4a$
$a5 \to 5a$
$a6 \to 6a$
$a7 \to 7a$
$a8 \to 8a$
$a9 \to 9a$
$0a \to 0b$
$1a \to 1b$
$2a \to 2b$
$3a \to 3b$
$4a \to 4b$
$5a \to 5b$
$6a \to 6b$
$7a \to 7b$
$8a \to 8b$
$9a \to 9b$
$\Lambda \to a$

Применим этот алгоритм к пустому слову Λ. Нетрудно понять, что на каждом шаге должна применяться самая последняя формула схемы. Получается бесконечный процесс:

$0b \to \cdot 1$	$8b \to \cdot 9$	$a5 \to 5a$	$3a \to 3b$
$1b \to \cdot 2$	$9b \to b0$	$a6 \to 6a$	$4a \to 4b$
$2b \to \cdot 3$	$b \to \cdot 1$	$a7 \to 7a$	$5a \to 5b$
$3b \to \cdot 4$	$a0 \to 0a$	$a8 \to 8a$	$6a \to 6b$
$4b \to \cdot 5$	$a1 \to 1a$	$a9 \to 9a$	$7a \to 7b$
$5b \to \cdot 6$	$a2 \to 2a$	$0a \to 0b$	$8a \to 8b$
$6b \to \cdot 7$	$a3 \to 3a$	$1a \to 1b$	$9a \to 9b$
$7b \to \cdot 8$	$a4 \to 4a$	$2a \to 2b$	$\Lambda \to a$

$$\Lambda \to \Lambda a \to \Lambda aa \to \Lambda aaa \to \Lambda aaaa \ldots$$

Это означает, что к пустому слову данный алгоритм неприменим.

Если применить этот алгоритм к слову 499, получим такую последовательность слов:

$$\Lambda 499 \to a499 \to 4a99 \to 49a9 \to 499a \to 499b \to 49b0 \to 4b00 \to 500.$$

Применим этот алгоритм к слову 328

$$\Lambda 328 \to a328 \to 3a28 \to 32a8 \to 328a \to 328b \to 329.$$

В рассмотренном примере нормальный алгоритм построен в алфавите B, являющемся существенным расширением алфавита A, но данный алгоритм *слова в алфавите* A перерабатывает снова в *слова в алфавите* A. В этом случае говорят, что **алгоритм задан над алфавитом** A.

Теорема 6.8. *Всякая частично рекурсивная функция является частично вычислимой по Маркову функцией*.

Следствие. *Если частичная функция вычислима по Маркову, то она частично рекурсивна, если же она всюду определена и вычислима по Маркову, то она рекурсивна*.

Представленные теорема и следствие устанавливают эквивалентность понятий вычислимости по Маркову и частичной рекурсивности.

Тезис Чёрча утверждает, в свою очередь, что понятие вычислимости эквивалентно понятию рекурсивности.

Тезис Чёрча. *Каждая вычислимая функция частично рекурсивна.*

Тезис Тьюринга. *Всякий алгоритм* (*в интуитивном понимании этого слова*) *может быть реализован некоторой машиной Тьюринга.*

А.А. Марков сформулировал соответствующий принцип, названный им **принципом нормализации**: *всякий алгоритм в A вполне эквивалентен относительно A некоторому нормальному алгоритму над A*.

Сформулированный принцип, как и тезисы Тьюринга и Чёрча, носит внематематический характер и не может быть строго доказан. Он выдвинут на основании математического и практического опыта.

Все, что в предыдущих параграфах было сказано о тезисах Тьюринга и Чёрча, можно с полным правом отнести и к принципу нормализации Маркова.

Теорема 6.9. *Всякая функция, вычислимая по Тьюрингу, будет также нормально вычислимой.*

Теорема 6.10. *Всякая нормально вычислимая функция вычислима по Тьюрингу.*

Теорема 6.11. *Следующие классы функций* (*заданные на натуральных числах и принимающие натуральные значения*) *совпадают:*

а) *класс всех функций, вычислимых по Тьюрингу;*

б) *класс всех частично вычислимых функций;*

в) *класс всех нормальных алгоритмов Маркова.*

Полезно уяснить смысл и значение этого важного результата.

В разное время в разных странах ученые независимо друг от друга, изучая интуитивное понятие алгоритма и алгоритмической вычислимости, создали теории, описывающие данное понятие, которые оказались равносильными.

Этот факт служит мощным косвенным подтверждением адекватности этих теорий опыту вычислений, справедливости каждого из тезисов Тьюринга, Чёрча и Маркова.

В самом деле, ведь если бы один из этих классов оказался шире какого-либо другого класса, то соответствующие тезисы Тьюринга и Чёрча или принцип нормализации Маркова были бы опровергнуты.

Например, если бы класс нормально вычислимых функций оказался бы шире класса рекурсивных функций, то существовала бы нормально вычислимая, но не рекурсивная функция.

В силу ее нормальной вычислимости она была бы алгоритмически вычислима в интуитивном понимании алгоритма, и предположение о ее нерекурсивности опровергло бы тезис Чёрча.

Но классы Тьюринга, Чёрча и Маркова совпадают, и такой функции не существует.

Можно отметить, что существуют и другие теории алгоритмов, и для всех них также доказана их равносильность с рассмотренными теориями.

Тренажер для изучения универсального исполнителя «Нормальные алгоритмы Маркова»
http://kpolyakov.spb.ru/prog/nma.htm

Урок 2. Простейшие примеры. Применение алгоритмов к словам (8:06)
https://www.youtube.com/watch?v=dWI9MokfoXk

Урок 3. Некоторые особенности работы нормальных алгоритмов Маркова (6:33)
https://www.youtube.com/watch?v=7FIf6sVrqqc

Урок 4. Примеры нормальных алгоритмов Маркова.

Часть 1 (7:12)
https://www.youtube.com/watch?v=xf2062PkAIM&t=36s

Часть 2 (5:30)
https://www.youtube.com/watch?v=AlHhRz9QCAk&t=134s

Часть 3 (3:53)
https://www.youtube.com/watch?v=KP63AtOjHGo&t=45s

Ответы к заданиям урока 4 (0:56)
https://www.youtube.com/watch?v=0JQu3C-8Vc4

Урок 5. Применимость нормальных алгоритмов Маркова (9:19)
https://www.youtube.com/watch?v=NsqGV4W_870&t=3s

Урок 6. Применимость нормальных алгоритмов Маркова. Примеры.

Часть 1 (8:17)
https://www.youtube.com/watch?v=XZgzjRsyf9M

Часть 2 (8:35)
https://www.youtube.com/watch?v=2U_YmEsob0k&t=387s