Министерство образования и науки Донецкой народной республики

Донецкая академия управления

Ю.Н. Добровольский

ТЕОРИЯ АЛГОРИТМОВ

Учебно-методическое пособие

Рекомендовано к изданию учебно-издательским советом
Донецкой академии управления

Донецк
Донецкая академия управления
2018

УДК 510.5(075.8)
ББК 22.161.1
М54

Рекомендовано к изданию учебно-издательским советом Донецкой академии управления

Рецензенты:
– Н. В. Брадул, канд. физ.-мат. наук, доцент, зав. кафедрой информационных технологий Донецкой академии управления;
– В. Н. Павлыш, д-р техн. наук, профессор, зав. кафедрой прикладной математики Донецкого национального технического университета.

Добровольский, Ю. Н.

М54 Теория алгоритмов: учебно-методическое пособие. Донецк: Донецкая академия управления, 2018. – 55 с.

Рассмотрены несколько формализаций понятия алгоритма: конечные автоматы, машины Тьюринга, нормальные алгоритмы Маркова и рекурсивные функции.
Изложение материала сопровождается большим количеством примеров и задач разного уровня сложности.
Предназначено для студентов университетов, технических вузов, обучающихся по направлениям «Прикладная математика» и «Прикладная информатика».

УДК 510.5(075.8)
ББК 22.161.1

СОДЕРЖАНИЕ

Лекция №1

Введение

Термин «алгоритм » или («алгорифм») происходит от имени арабского математика Мохаммада аль- Хорезми, жившего в IX веке н. э. В своем трактате «Китаб аль-джебр валь-мукабала» (по арабски «Книга о сложении и вычитании») он впервые дая описание, придуманной еще в Индии десятичной позиционной системы счисления. Тогда же была впервые введена цифра «0» для обозначения пропущенной позиции в записи числа.

В Европу учение аль-Хорезми пришло в XII веке и называлось «Algoritmi de numero Indorum» («Алгоритмы о счёте индийском»). Латинский перевод книги начинается словами «Dixit Algorizmi» («Сказал Алгоризми»). Таким образом, имя автора стало нарицательным, и средневековые математики под «алгоритмами» стали понимать правила выполнения четырех основных арифметических действий над числами в десятичной системе счисления.

В XVII веке немецкий математик Г. Лейбниц использовал термин алгоритм как системный способ решения проблем дифференцирования и интегрирования.

В 30-е годы XX века появились первые работы по уточнению понятия алгоритма и его изучению математиками А. Тьюрингом, Э. Постом, К. Геделем, А. А. Марковым, А. Черчем. Было сформулировано несколько формальных определений понятия алгоритма, но потом выяснилось, что все они равносильны и определяют одно и то же.

Дальнейшее развитие термина алгоритм возникло при появлении электронно-вычислительных машин и персональных компьютеров, связано с бурным развитием информатики.

Алгоритм – набор четких недвусмысленных инструкций, следуя которым можно по входным данным определенного вида получить на выходе некоторый результат.

Примеры: алгоритм сложения натуральных чисел столбиком; вычисление наибольшего общего делителя двух натуральных чисел (алгоритм Евклида); построение треугольника по трем сторонам с помощью циркуля и линейки; дифференцирование многочлена; поиск слова в двуязычном переводном словаре и т. п.

Алгоритмы содержат ряд общих требований (свойств):

1. Дискретность – алгоритм должен осуществляться путем выполнения последовательности элементарных действий.

2. Детерминированность (определенность) – в каждый момент времени, следующий шаг работы алгоритма однозначно определяется состоянием системы.

3. Результативность – алгоритм должен завершать работу и выдавать результат за конечное число действия.

4. Массовость – возможность применять алгоритм к разным наборам входных данных.

5. Понятность – алгоритм должен включать только те инструкции, которые доступны исполнителю.

Теория алгоритмов – часть математической логики, изучающая общие свойства и закономерности алгоритмов, а так же разнообразные формальные модели и их представление. Теория алгоритмов предназначена для исследования проблемы разрешимости и обоснования математики.

Все вопросы, касающиеся теории алгоритмов, можно условно разбить на четыре категории:

1. Вопросы, относящиеся к теории вычислимости (какие функции являются вычислимыми, какие нет).

2. Вопросы, относящиеся к теории сложности, т. е. вопросы, которые позволяют выяснить, за какое время можно вычислить ту или иную функцию, а также вопросы поиска оптимального решения.

3. Вопросы, относящиеся к программированию, т. е. реализации алгоритма.

4. Вопросы, относящиеся к методам вычислений, т. е. вопросы о нахождении приближенных решений уравнений, неравенств и их систем (численные методы).

Основной объем этого курса – вопросы 1-й категории. Далее мы рассмотрим несколько формализаций понятия алгоритма: конечные автоматы, машины Тьюринга, нормальные алгоритмы Маркова и рекурсивные функции.

1. Конечные автоматы

Алфавит – не пустое конечное множество элементов, которые называются символами или буквами. Например, кириллица {а,б,…,я}, латиница {a,b,…,z}, цифры десятичной системы счисления {0,1,…,9} или двоичной {0,1}, произвольные конечные множества {1}, {a,b}, {*,0,1}, {-,+}.

Словом алфавита $\sum$ назовем любую конечную последовательность символов из $\sum$. Например, для алфавита {0,1,…,9} словом будет являться любое натуральное число, в алфавите {a,b} можно выделить слова a, aaa или abba. Слова в алфавите $\sum$ будем обозначать греческими буквами **α**, **β**, **γ** и т. д.

Введем понятие **пустого слова**, как слова без букв. Для любого алфавита $\sum$ пустое слово **e** есть слово в этом алфавите. Множество всех слов алфавита $\sum$ обозначим $\sum^*$.

Каждое слово имеет длину, равную числу символов в этом слове. Длина слова **α** обозначается через **|α|**. Например, в алфавите {a,b} длина слов |a|=1, |ababa|=5. Длина пустого слова равна нулю, т. е. |e|=0. Для слова α и натурального числа $1 \le i \le |\alpha|$ через $\boldsymbol{\alpha_i}$ обозначим i-тую букву в этом слове.

Например, если **α=abbaa**, то $\boldsymbol{\alpha_2}$**=b** или $\boldsymbol{\alpha_4}$**=a**.

На множестве слов $\sum^*$ введем бинарную алгебраическую операцию **конкатенация (склеивание)**. Конкатенация слов α и β есть слово, в котором сначала идут символы слова α, а затем все символы слова β. Конкатенацию слов α и β обозначим $\alpha\cdot\beta$. Например, если в алфавите {a,b} α=abb и β=ba, то $\alpha\cdot\beta$=abbba, а $\beta\cdot\alpha$=baabb. В алфавите {0,1}: $01\cdot101=01101$. Операция конкате-

нации слов, подобно операции умножения чисел, порождает операцию **возведения в степень:** $a^n = \underbrace{a \cdot a \cdot \ldots \cdot a}_{n}$,

Слово в нулевой степени есть пустое слово: $a^0 = e$.

Конкатенация слов обладает следующими **свойствами:**

1. Ассоциативность. $\forall \alpha, \beta \in \sum^* \ \alpha \cdot (\beta \cdot \gamma) = (\alpha \cdot \beta) \cdot \gamma$;
2. Некоммутативность $\forall \alpha, \beta \in \sum^* \ \alpha \cdot \beta \neq \beta \cdot \alpha$;
3. Существование нейтрального элемента.

$\exists e \in \sum \ \forall \alpha, \beta \in \sum^* \ \alpha \cdot e = e \cdot \alpha = \alpha$.

Следовательно, множество слов $\sum^*$ образует алгебраическую структуру называемую **моноидом**.

Языком L алфавита $\sum$ называется непустое подмножество множества слов $\sum^*$ этого алфавита, т. е. $L \subseteq \sum^*$.

Конечным детерминированным автоматом называется структура вида

$$A = \langle \sum, Q, \delta, q_0, F \rangle. \qquad (1)$$

Множество $\sum$ называется **алфавитом автомата** A. Множество

$Q = \{q_0, q_1, \ldots, q_n\}$ называется **множеством состояний** автомата A, элементы q_i, $1 \leq i \leq n$ этого множества – состояниями. Состояние $q_0 \in Q$ называется **начальным состоянием**, с этого состояния автомат A начинает свою работу. Состояние q_0 входит в структуру (1), это говорит о том, что множество Q не может быть пустым. Множество $F \subseteq Q$ называется **множеством конечных состояний** автомата A, а его элементы конечными состояниями. **Функция переходов** δ: $Q \times \sum \rightarrow Q$ указывает на переход автомата из одного состояния в другое под действием символа из алфавита $\sum$.

Дадим строгое определение функции δ: для каждого $q \in Q$ и $a \in \sum$ существует единственное состояние $q' \in Q$, такое, что $q' = \delta(q,a)$ есть переход автомата из состояния q в состояние q′ под действием символа a.

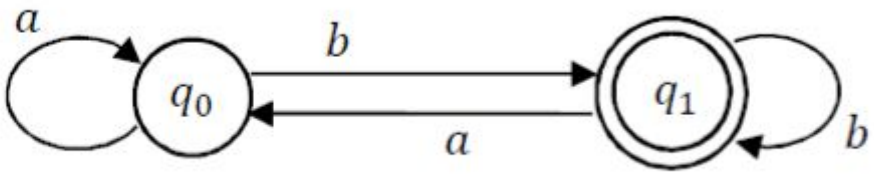
Например, пусть автомат A задан алфавитом $\sum = \{a,b\}$ и множеством состояний $Q = \{q_0, q_1\}$. Множество конечных состояний состоит из одного элемента и равно $F = \{q_1\}$. Функцию переходов представим таблицей:

$\delta(q,a)$	a	b
q_0	q_0	q_1
q_1	q_0	q_1

По таблице видно, что автомат под действием символа a переходит или остается в состоянии q_0; в состояние q_1 автомат попадает под действием символа b.

Конечный автомат для удобства восприятия принято обозначать в виде диаграммы (графа). Кружочками (вершинами графа) отмечены состояния автомата, стрелками (дугами) – переходы между состояниями. Конечные состояния обводятся двойными кружками.

Автомат из примера выше можно представить следующей диаграммой:



Таким образом, диаграмма позволяет однозначно определить автомат, тем самым исключается необходимость задавать каждый элемент структуры (1).

Для конечного автомата A определим функцию ρ: Q x ∑* → Q, которая переводит одно состояние автомата в другое под действием слова из ∑*. Для каждого q ∈Q и α ∈∑* значение ρ(q,α) определяется индукцией по длине слова α согласно следующей схеме:

1. ρ(q,e)=q;
2. ∀α ∈∑* и ∀a ∈∑ ρ(q,αa) = δ(ρ(q,α),a).

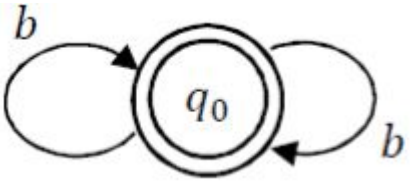
Значение ρ(q,α) равно состоянию, в которое перейдет автомат из состояния q, последовательно читая символы слова α и осуществляя переходы согласно функции δ.

Например, для автомата, изображенного на рисунке выше, прочитавслово abababиз начального состояния, автомат перейдет в состояние q_1, т. е. $\rho(q_0,ababab) = q_1$. Аналогично $\rho(q_0,aabba) = q_0$ или $\rho(q_1,abaa) = q_0$.

Автоматным языком L(A) или языком, распознаваемым автоматом A, называется множество таких слов, после прочтения, которых автомат из начального состояния попадает в конечное, т. е. L(A)={α∈∑*| ρ(q_0,α)∈F}. Следовательно, автомат – это алгоритм, который, получив на входе слово α, позволяет установить принадлежность α ∈L(A).

Автомат из примера выше распознает все слова алфавита ∑ = {a,b}, которые оканчиваются символом b, т. е. L(A) = {αb| α ∈∑*}.

Автомат A, который распознает все слова алфавита ∑ = {a,b}, т. е. L(A)=∑*, представлен на диаграмме:



1.1. С помощью диаграммы изобразите автомат, который в алфавите ∑ = {a,b} распознает:

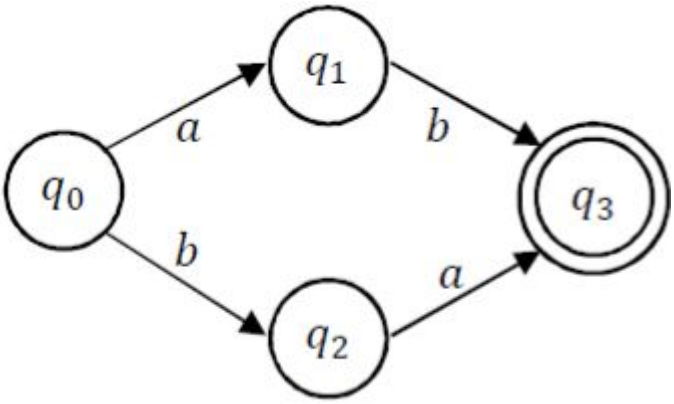
a) только слова ab и ba;

b) слова, начинающиеся символом a и оканчивающиеся символом b;

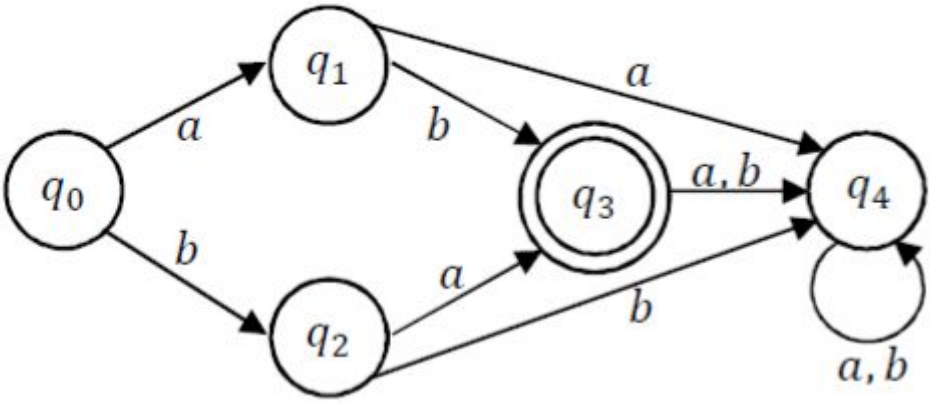
c) слова, состоящие только из символов a или только из символов b (a, aa, aaa,…, b, bb, bbb,…).

Решение. a) Построим диаграмму автомата, который читает только двухбуквенные слова ab и ba. После прочтения первого символа автомат

должен попасть в разные состояния (q_1 и q_2), так как должен запомнить, какой символ был прочтен ранее. После прочтения второго символа автомат должен оказаться в состоянии остановки (q_3). Итак, имеем первоначальное изображение искомого автомата:



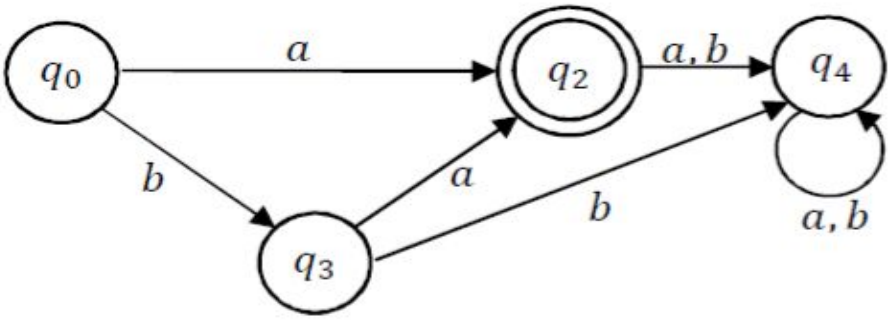
Состояния q_1 и q_2, изображенного автомата не удовлетворяют условиям детерминированности; так, например, из состояния q_1 неизвестно, куда попадет автомат после прочтения символа a. Аналогично для состояния q_2 и прочтения символа b из этого состояния. Для решения этой проблемы добавим новое состояния q_4, куда будет уходить автомат после прочтения ненужных символов. Это состояние необходимо зациклить по всем символам алфавита. В результате искомый автомат представим диаграммой:



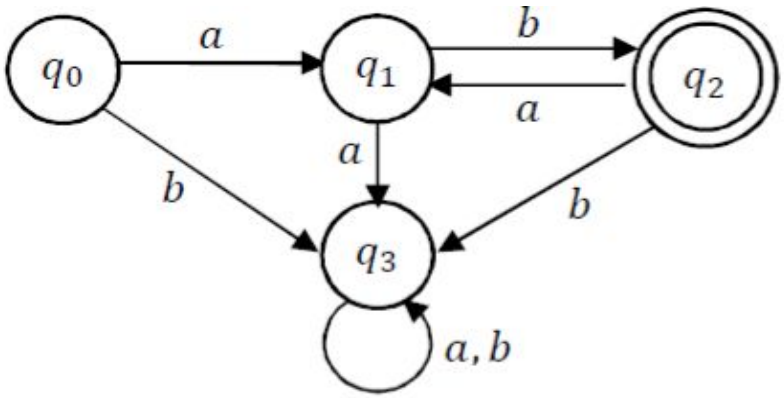
По рисунку видно, что попадая в состояние q_4, автомат никогда уже не будет в заключительном состоянии q_3, при этом требование детерминированности будет выполнено.

1.2. Для автомата, изображенного на диаграмме, определите автоматный язык:

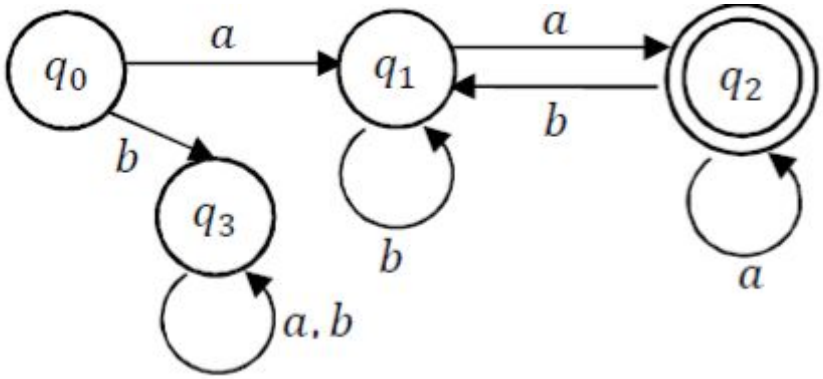
а)



b)



c)



Равносильность конечных автоматов. Два детерминированных конечных автомата называются **равносильными,** если совпадает их автоматный язык. Оптимальным считается тот из равносильных автоматов, который состоит из меньшего числа состояний.

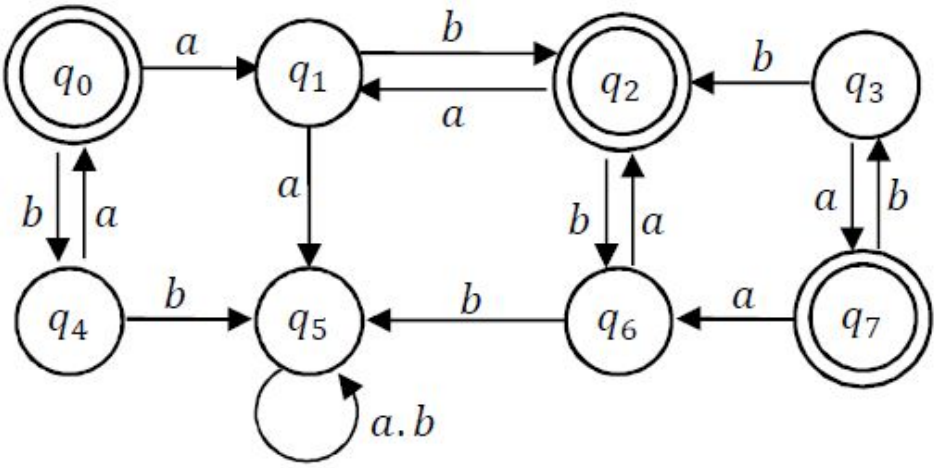
Пусть дан детерминированный конечный автомат A= <∑, Q, δ, q_0, F>. Опишем алгоритм поиска автомата, равносильного данному, с меньшим числом состояний.

Сначала выделим те состояния автомата, которые достижимы из начального q_0. Состояние называется **недостижимым,** если в него нельзя попасть в результате распознавания любого слова алфавита ∑. Обозначим достижимые состояния за $Q' \subseteq Q$. Затем множество этих состояний необходимо разбить на непустые непересекающиеся классы.

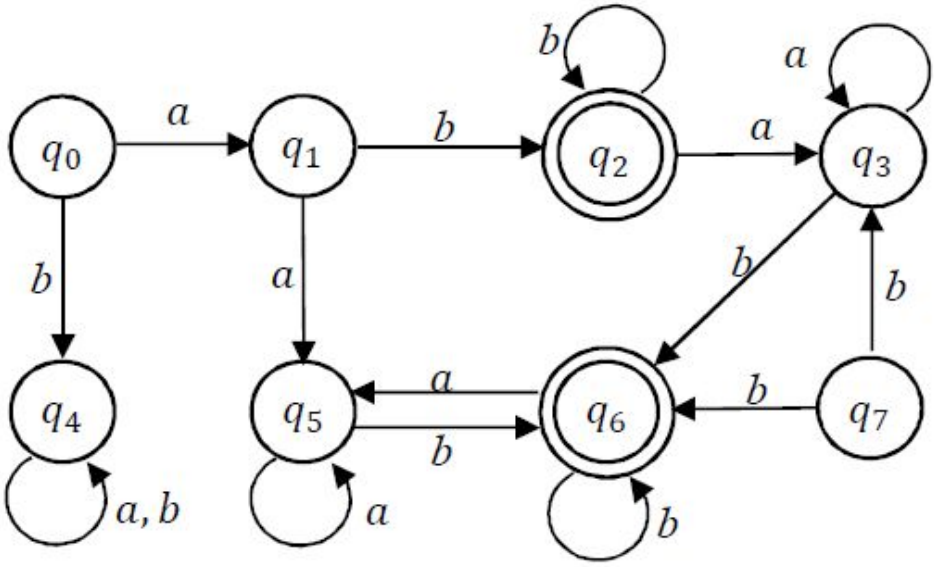
Для этого определим отношение эквивалентности, заданное на множестве Q': для $n \in N_0$ два достижимых состояния q и p находятся в одном классе, т. е. $q \approx_n p$ тогда и только тогда, когда $\forall \alpha \in \sum^* |\alpha| \leq n\ \rho(q,\alpha) \in F \leftrightarrow \rho(p,\alpha) \in F$. Таким образом, каждое из отношений $\approx_0, \approx_1, \ldots, \approx_n$ осуществляет разбиение множества Q'. Очевидно $\approx_0 \supseteq \approx_1 \supseteq \ldots \supseteq \approx_n$, причем в силу конечности Q' найдется такой номер n, при котором $\approx_n = \approx_{n+1}$ и процесс разбиения заканчивается. В результате, состояния, взятые по одному представителю от каждого класса эквивалентности из $\approx_n$, образуют множество состояний нового автомата A_1, равносильного данному A, но с меньшим числом состояний.

1.3. Построить автомат, равносильный представленному на диаграмме, с меньшим числом состояний.

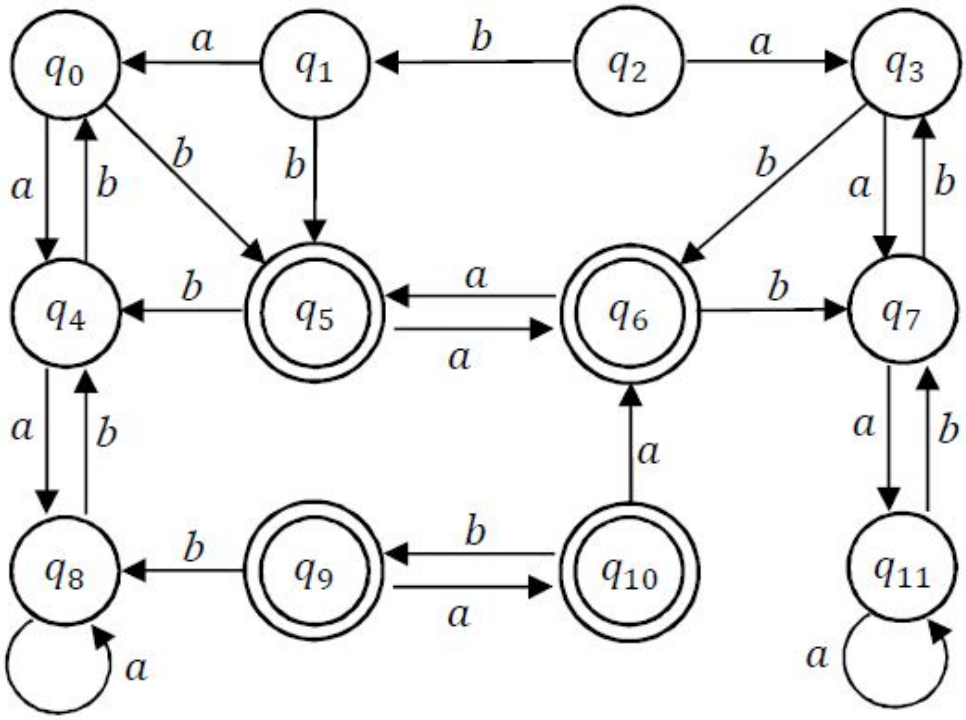
a)


q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
a
b
$a.b$

b)


q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
a
b
a, b

c)


q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
q_8
q_9
q_{10}
q_{11}
a
b

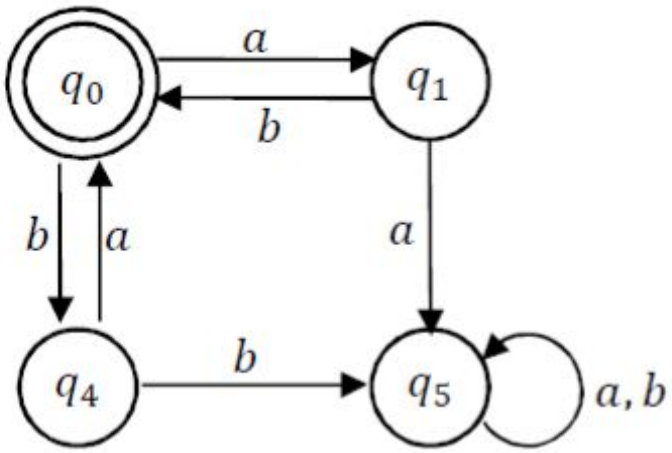
Решение. а) По рисунку видно, что недостижимыми являются состояния q_3 и q_7, в которые невозможно попасть, читая слова из начального состояния q_0. Множество Q' будет равно $\{q_0,q_1,q_2.q_4,q_5,q_6\}$.

Осуществим разбиение множества Q' согласно отношению эквивалентности $\approx_0$. Так как только пустое слово e имеет длину равную нулю, то два класса $q \approx_0 p$ тогда и только тогда, когда $\rho(q,e) \in F \leftrightarrow \rho(p,e) \in F$. Следовательно, отношение $\approx_0$ имеет два класса и фактор-множество $Q'/_{\approx 0}$ можно записать как $\{q_0,q_2\} \cup \{q_1,q_4,q_5,q_6\}$.

Далее, так как $\approx_0 \supseteq \approx_1$, то разбиению можно подвергать только классы из $\approx_0$. Подвергнем разбиению класс $\{q_0,q_2\}$, читая однобуквенные слова алфавита {a,b}. Имеем $q_0 \approx_1 q_2$, так как $\rho(q_0,a)= \rho(q_2,a) = q_1$ и $\rho(q_0,b) = q_4 \approx_0 q_6 = \rho(q_2,b)$. Следовательно, класс $\{q_0,q_2\}$ дальнейшему разбиению можно не подвергать. Для состояний из второго класса $\{q_1,q_4,q_5,q_6\}$ имеем: $\rho(q_4,b)= \rho(q_6,b)=q_5$ и $\rho(q_4,a) = q_0 \approx_0 q_2 = \rho(q_6,a)$, в итоге $q_4 \approx_1 q_6$ и эти состояния должны находиться в одном классе. Для состояния q_1: $\rho(q_1,a) = q_5 \neq_1 q_0 = \rho(q_4,a)$ и, значит q_1 не лежит в одном классе с q_4 и q_6. Наконец, $\rho(q_1,b)=q_2$ и $\rho(q_5,b)=q_5$, что указывает на принадлежность состояний q_1 и q_5 к разным классам. Таким образом, отношение $\approx_1$ имеет 4 класса эквивалентности: $\{q_0,q_2\}$, $\{q_1\}$, $\{q_4,q_6\}$, $\{q_5\}$.

Для поиска $\approx_2$ осталось проверить пару состояний q_4 и q_6. Имеем $\rho(q_4,aa) = \rho(q_6,aa) = q_1$, $\rho(q_4,bb) = \rho(q_6,bb)=q_5$, $\rho(q_4,ab) = q_4 \approx_1 q_6=\rho(q_6,ab)$ и $\rho(q_4,ba) = \rho(q_6,ba) = q_5$. Значит $q_4 \approx_2 q_6$ и эквивалентности $\approx_1$ и $\approx_2$ совпадают. Дальнейшее разбиение классов невозможно.

Определим состояния нового автомата A_1, равносильного данному, с меньшим числом состояний. В множество состояний Q_1 этого автомата возьмем по представителю из каждого класса эквивалентности $\approx_2$. Например, $Q_1=\{q_0,q_1,q_4,q_5\}$. Изображение автомата A_1 представлено на диаграмме:



Конечным недетерминированным автоматом называется структура вида

$$A = < \sum, Q, \Delta, q_0, F > \qquad (2)$$

Множества $\sum$, Q, F и состояние q_0 те же, что и в конечном детерминированном автомате (1). Тройки $\Delta=\{(q,a,q')|q,q'\in Q,a\in\sum\}$ образуют **отношение переходов,** согласно которым автомат осуществляет переход из состояния q в состояние q′ под действием символа a. Отношение Δ не является функцией, что указывает на возможность осуществить переход из одного состояния в другое под действием разных символов алфавита $\sum$, при этом не для каждых $q \in Q$ и $a \in \sum$ переход существует. Тройку (q, a, q′) называют **скачком** из q в состояние q′.

Недетерминированные автоматы, также как и детерминированные, изображают в виде диаграмм, вершины которого есть состояния, а дуги отношение переходов. На диаграмме детерминированного автомата из каждого состояния выходит столько дуг, сколько символов содержит алфавит. В то время на диаграмме недетерминированного автомата из состояния может выходить любое количество дуг, в том числе и ни одной. Кроме того, появятся дуги, помеченные символами e пустого слова. Заметим, что детерминированные автоматы есть частный случай недетерминированных.

Недетерминированные автоматы в ходе работы читают символы подаваемого на вход слова. Если в некоторый момент времени автомат находится в состоянии q и под действием одного и того же символа может перейти в несколько состояний, то он выбирает любое и переходит в него. Если таких состояний нет, то автомат останавливает работу и не читает дальнейшие символы слова. Кроме того, автомат может совершить скачок, если он возможен в текущем состоянии. Таким образом, можно считать, что недетерминированный автомат обладает «интуицией»: находясь перед выбором состояний, он совершает наилучший переход.

Недетерминированные автоматы, аналогично детерминированным, служат для распознавания слов. Недетерминированный автомат распознает слово, если после прочтения его из начального состояния q_0 он попадает в состояние из F.

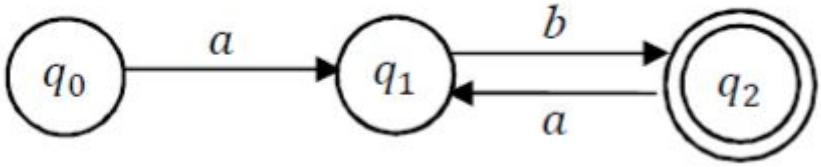
Языком недетерминированного автомата L(A) называется множество слов, распознаваемых автоматом A.

Утверждение. Язык распознается некоторым детерминированным автоматом тогда и только тогда, когда он распознается некоторым недетерминированным автоматом.

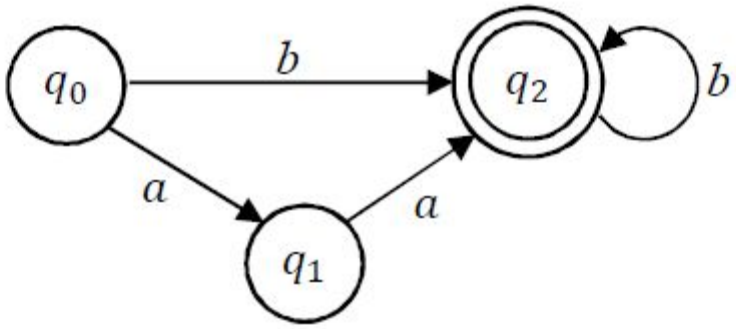
Таким образом, можно говорить просто о языках, распознаваемых конечными автоматами, не уточняя, какие автоматы имеются в виду. Причем диаграмма недетерминированного автомата будет выглядеть более компактно, чем диаграмма детерминированного, распознающего один и тот же язык. Например, в диаграмме детерминированного автомата можно убрать состояния и дуги, из которых и благодаря которым нельзя осуществить переход в конечные состояния F.

1.4. Опишите язык недетерминированного автомата A, изображенного на диаграмме:

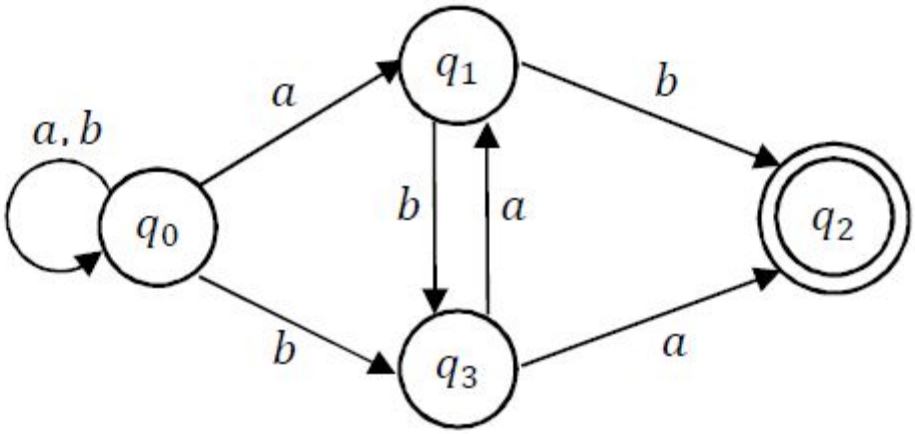
a)



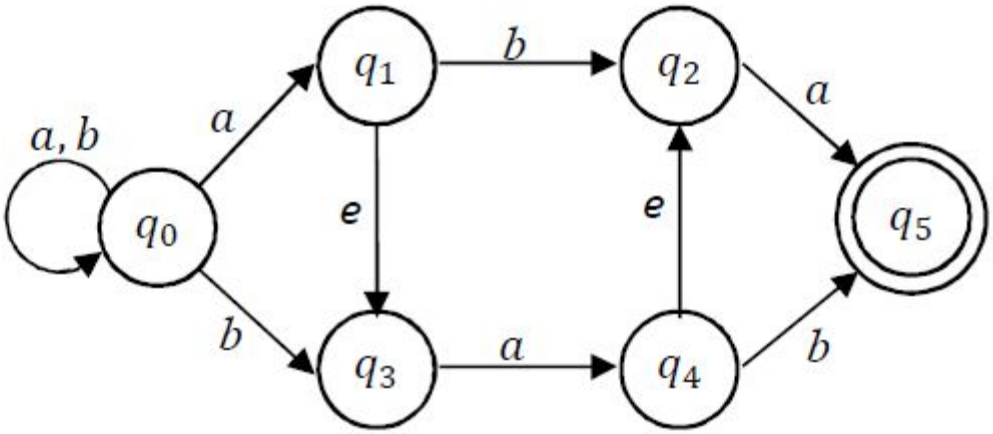
b)



c)



d)



Решение. c). Недетерминированный автомат A, изображенный на рисунке, распознает слова алфавита $\sum$={a,b}. В заключительное состояние q_2 автомат попадает, как минимум, прочитав слова ab и ba. Перед этим автомат может прочитать любое слово, значит, язык данного недетерминированного автомата содержит слова, оканчивающиеся символами ab или ba. Язык автомата можно записать, указав его характеристическое свойство: $L(A)=\{\alpha ab,\alpha ba|\alpha \in \sum^*\}$.

Операции над языками. Регулярные языки. Дадим альтернативное описание автоматных языков, независящее от понятия конечного автомата.

Для этого введем ряд операций над языками. Напомним, что языки некоторого алфавита $\sum$ есть подмножество $\sum^*$, то есть множества, к которым применимы операции алгебры множеств.

Для этого введем ряд операций над языками. Напомним, что языки некоторого алфавита $\sum$ есть подмножества $\sum^*$, то есть множества, к которым применимы операции алгебры множеств. В частности, **объединением языков** L_1 и L_2 называется язык, состоящий из слов, которые входят в L_1 или L_2. Или формально $L_1 \cup L_2 = \{\alpha \in \sum^* | \alpha \in L_1$ или $\alpha \in L_2\}$.

Кроме теоретико-множественных операций введем еще две, применимые только к языкам.

Конкатенацией языков L_1 и L_2 называется язык $L_1 \cdot L_2 = \{\alpha \cdot \beta \in \sum^* | \alpha \in L_1$ и $\beta \in L_2\}$, состоящий из слов, которые будут всевозможными конкатенациями слов языка L_1 со всеми словами языка L_2.

Например, если в алфавите $\sum = \{a,b\}$ языки $L_1 = \{a,bb\}$ и $L_2 = \{b,aa\}$ то язык $L_1 \cdot L_2$ содержит слова {ab,bbb,aaa,bbaa}. Заметим, что конкатенация языков не обладает свойством коммутативности.

Операцию **навешивание звездочки Клини** введем индуктивно. Пусть L – язык алфавита $\sum$. Для каждого натурального n определим язык L^n:

1. $L^0 = \{e\}$
2. $L^{n+1} = L^n \cdot L$

Тогда $L^* = L^0 \cup L^1 \cup \ldots \cup L^n \cup \ldots$. Говорят, что язык L^* получен из языка L навешиванием звездочки Клини. Например, для языка $L = \{a\}$ язык L^* состоит из всевозможных конкатенаций символа a. Заметим, что множество L^* будет бесконечным и обязательно содержит пустое слово e. Навешивание звездочки Клини на язык $L = \{ab,ba\}$ будет состоять из множества слов $L^* = \{e,ab,ba,abab,abba,baba,baab,\ldots\}$. Кстати, обозначение множества слов $\sum^*$ алфавита $\sum$ имеет звездочку Клини.

Регулярным выражением алфавита $\sum$ называется выражение, составленное из символов этого алфавита при помощи операций объединения, конкатенации и навешивания звездочки Клини. Например, каждое из регулярных выражений $a(a \cup b)^*b$, $ab^* \cup ba^*$ или $(a \cup ba)b^*$ порождает свой регулярный язык.

Например, язык регулярного выражения $a(a \cup b)^*b$ состоит из всех слов алфавита $\sum = \{a,b\}$, начинающихся символом a и заканчивающихся символом b.

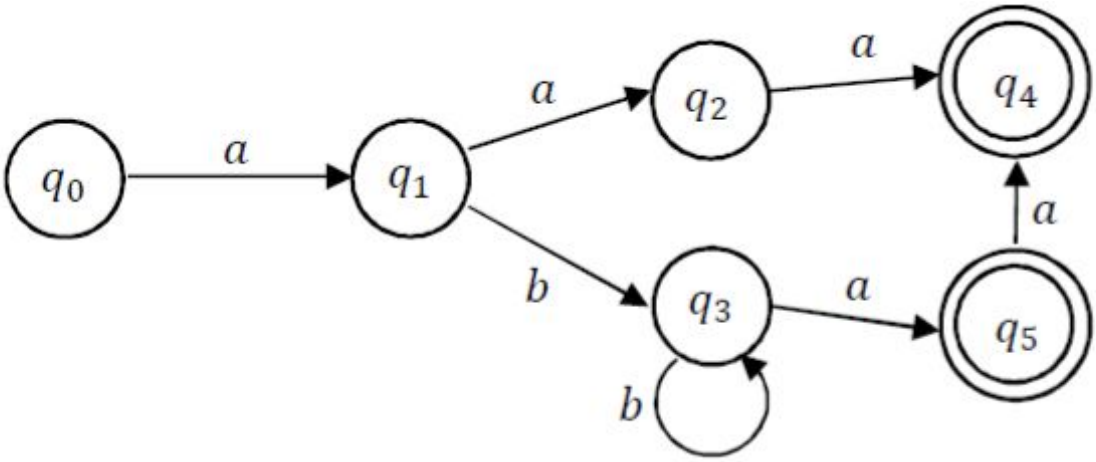
Заметим, что множество всех слов алфавита $\sum = \{a,b\}$ можно также представить в виде регулярного выражения $\sum^* = (a \cup b)^*$.

Утверждение. Язык является регулярным тогда и только тогда, когда он распознается некоторым конечным автоматом.

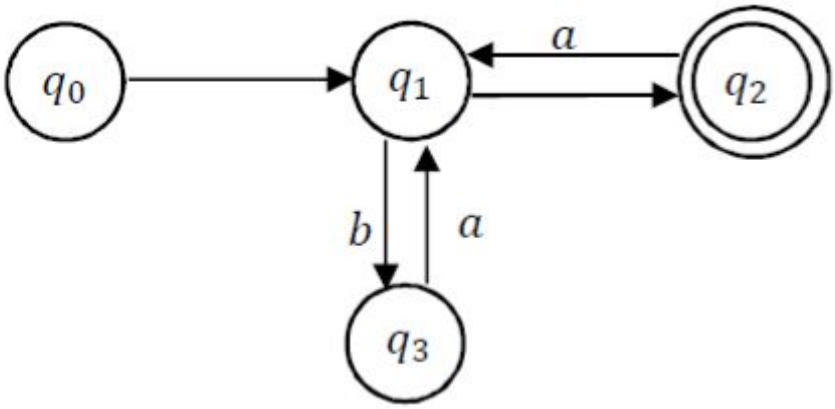
Значит, любой конечный автомат, как устройство распознающее языки, можно представить не только диаграммой, но и регулярным выражением. И, наоборот, для каждого регулярного выражения существует конечный автомат, который распознает его язык.

Например, язык автомата, который распознает все слова алфавита, оканчивающиеся символом b, легко представить регулярным выражением $L(A)=(a \cup b)^* b$.

Пусть регулярный язык задан выражением $L(A)=ab^*(a \cup b)a$. Автомат, который распознает этот язык:



Для регулярного выражения $a(ab \cup ba)^* b$ автомат можно представить диаграммой:



1.5. Для языков L_1 и L_2 алфавита {a,b} найдите: $L_1 \cup L_2$, $L_1 \cdot L_2$, $L_2 \cdot L_1$, L_1^*, L_2^*.

a) L_1={a} и L_2={b};

b) L_1={a} и L_2={a,bb};

c) L_1={ab,ba} и L_2={a,b};

d) L_1={aab,baa} и L_2={a,b,ab,ba}.

Решение. b) Пусть даны два языка: L_1={a} и L_2={a,bb}. Объединение языков L_1 и L_2 есть язык, содержащий как слова из L_1, так и из L_2. Здесь это два слова: $L_1 \cup L_2$={a,bb}.

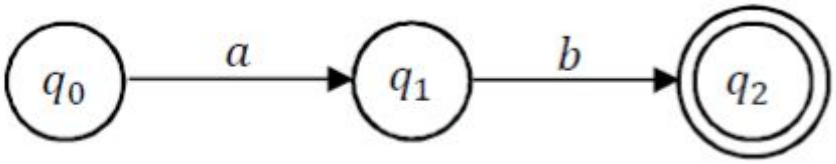
Выпишем всевозможные конкатенации слов из языков L_1 и L_2. Получим: $L_1 \cdot L_2$ = {aa,abb}, $L_2 \cdot L_1$={aa,bba}.

Звездочка Клини языка L_1={a} содержит все конкатенации единственного однобуквенного слова a: L_1^* = {e,a,aa, …}. Выпишем все конкатенации слов языка L_2, получим L_2^*={e,a,bb,aa,abb,bba,aaa,bbbb,…}.

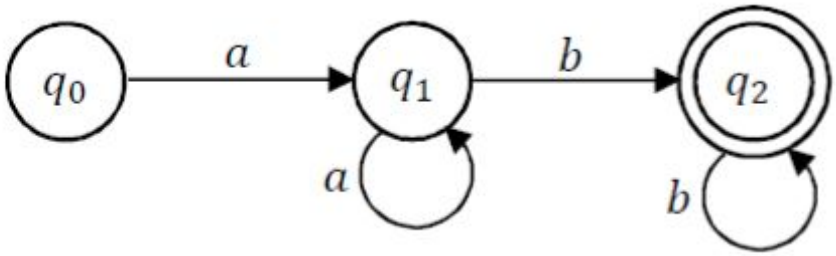
1.6. Постройте автомат, который распознает язык, заданный регулярным выражением в алфавите {a,b}:

a) aa^*bb^*;
b) $a^*b^*a^*b^*$;
c) $a(a^* \cup b^*)b$;
d) $(ab)^* \cup (ba)^*$;
e) $a(a \cup b)^*b$;
f) $a(ab \cup ba)^*ab$;
g) $(ab \cup ba)a^*b$.

Решение. a) Регулярное выражение aa^*bb^* содержит слова {ab,aab,abb,aabb,a…ab…b,…}, которые начинаются символами a и заканчиваются символами b. Автомат обязательно должен распознать слово минимальное длины ab, именно с этого начинаем строить автомат:



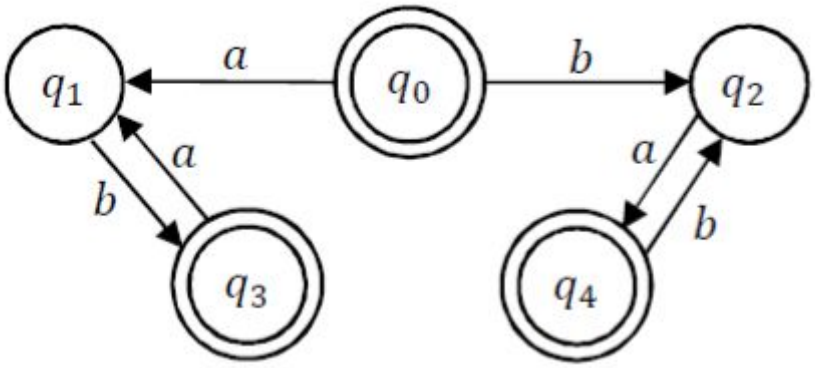
Звездочки Клини над символами a и b в регулярном выражении изобразим петлями на состояниях q_1 и q_2 соответственно. В результате получим изображение искомого автомата.



Заметим, что представлена диаграмма недетерминированного конечного автомата в силу того, что он более компактен, чем соответствующий ему детерминированный.

Внимание! Искомый автомат должен распознавать слова только регулярного выражения, распознавание других слов не допустимо.

d) Регулярное выражение $(ab)^* \cup (ba)^*$ содержит слова, которые представляют собой либо последовательности ab, либо последовательности ba. Данный автомат распознает пустое слово e, так как оно содержится в любом языке при навешивании звездочки Клини. Таким образом, начальное состояние q_0 обязано быть и заключительным. Искомый автомат будет представлен диаграммой:



Конечные автоматы – один из способов формализации понятия алгоритма, который способен ответить лишь на вопрос распознавания слов множества $\sum^*$ с двумя возможными ответами {да, нет}. Далее мы рассмотрим три популярные формализации понятия алгоритма с более широкими возможностями.

2. НОРМАЛЬНЫЕ АЛГОРИТМЫ МАРКОВА

Нормальные алгоритмы (или алгорифмы) разрабатывались в конце 1940-х годов XX века советским математиком А.А. Марковым. Алгоритмы Маркова представляют собой набор правил по переработке слов некоторого алфавита.

Пусть дан алфавит $\sum$ и слова в этом алфавите. Слово α является вхождением слова β, если существуют такие слова γ_1 и γ_2 (возможно пустые), что $\beta = \gamma_1\alpha\gamma_2$. Например, в алфавите $\sum$={a,b} слова bbaa и abba являются вхождениями в слово abbaaab. В дальнейшем под вхождениями слова в данное слово будем понимать только первые вхождения. Пустое слово является первым вхождением любого слова.

Марковской подстановкой (или просто подстановкой) называется выражение вида $\alpha \rightarrow \beta$, которое осуществляет замену слова α на слово β. Подстановки применяются для преобразования слов алфавита $\sum$.

Пусть дано слово γ. Подстановка $\alpha \rightarrow \beta$, примененная к слову γ, означает замену первого вхождения слова α в γ на слово β. Полученное слово называют результатом подстановки $\alpha \rightarrow \beta$ к слову γ. Если не существует вхождение слова α в γ, то подстановка считается неприменимой.

Например, пусть в алфавите $\sum$={a,b} дано слово abbab. В результате применения подстановки $ab \rightarrow ba$ к слову abbab будет получено слово babab, результатом подстановки $abb \rightarrow e$ (e – пустое слово) будет слово ab. Подстановка $abab \rightarrow aa$ не применима к слову abbab.

Некоторые применения марковских подстановок к словам различных алфавитов представлены в таблице.

Алфавит	Преобразуемое слово	Подстановка	Результат подстановки
латиница	**b**ook	b → c	**c**ook
кириллица	оля	e → k	коля
0,1,…,9	1**23**5230	23 → 7	175230
0,1	**1**0011	1→ 010	**010**0011
0,1,…,H	2D1**3A**	3A → e	2D1
ASCII	a12b23	a123 → sad	a12b23

В последней строке подстановка a123 → sad оказалась не применима для данного слова в кодировке ASCII (American standard code interface interchange).

2.1. Пусть для слов в алфавите ∑={a,b} заданы следующие марковские подстановки:

a) a → b; d) aab → a; g) aba → bab;
b) ab → ba; e) babb → e; h) baab → ababa;
c) e → b; f) abaa → bb; i) bbb → a.

Примените каждую из них к слову abaababbb. Примените каждую подстановку к данному слову максимальное число раз, если это возможно и не приведет к зацикливанию.

Решение. a) Для применения марковской подстановки a → b к слову необходимо обнаружить первое вхождение правой части подстановки (в нашем случае это символ a) в слове abaababbb и выполнить замену словом в правой части подстановки (символ b). Результатом от однократного применения подстановки a → b к слову **a**baababbb будет слово bbaababbb.

Подстановку можно применить еще несколько раз, обнаруживая первое вхождение символа a и выполняя замену на символ b. Последовательность получающихся слов представим следующим образом bb**a**ababbb, bb**b**ababbb, bbb**b**babbb, bbbbb**b**bbb.

b) Результатом однократного применения подстановки ab → ba к слову abaababbb будет слово **ba**aababbb. Дальнейшее применение приведет к результату: baa**ba**abbb, ba**ba**aabbb, b**ba**aaabbb,…, bbbbbaaaa.

c) Однократное применение подстановки e → b (e – пустой символ) приведет к добавлению символа b в начале слова: **b**abaababbb. Последующее применение данной подстановки приведет к зацикливанию, т.е. многократному добавлению символа b в начале слова.

2.2. Пусть для слов в алфавите ∑={0,1} заданы следующие марковские подстановки:

a) 101 → 10; c) 00 → 11; e) 0110 → 11;
b) 10 → e; d) 10 → 101; f) 1100 → 01.

Примените каждую из них к слову 1011001100. Примените каждую подстановку к данному слову максимальное число раз, если это возможно и не приведет к зацикливанию.

Решение. a) Результатом однократного применения подстановки 101 → 10 к слову 1011001100 будет слово 101001100. Дальнейшее применение приведет к результату: 10001100.

b) Однократное применение подстановки 10 → e приведет к удалению первого вхождения символов 10: 11001100. Последующее применение данной подстановки приведет к пустому слову: 101100, 1100, 10, e.

Марковская подстановка вида α → **.**β называется **заключительной**, если после ее применения работа нормального алгоритма прекращается.

Нормальным алгоритмом Маркова (далее НАМ) называется структура вида

$$NA = < \sum, \sum', S >, \qquad (3)$$

где ∑ - алфавит, слова которого перерабатывает алгоритм, расширенный алфавит ∑' содержит ∑ и вспомогательные символы, которые могут быть включены в марковские подстановки. Элемент S структуры (3) называется **схемой алгоритма.** Схема S представляет собой упорядоченный конечный набор подстановок.

$$\begin{cases} \alpha_1 \to \tau_1 \beta_1 \\ \alpha_2 \to \tau_2 \beta_2 \\ \cdots\cdots\cdots \\ \alpha_k \to \tau_k \beta_k \end{cases}$$

τ_i (i =1,2,…,k) есть "**.**" для заключительной подстановки или пустой символ для обыкновенной.

Нормальный алгоритм Маркова **вычисляет словарную функцию** f: $\sum^* \to \sum^*$. Под **вычисление слова γ** алфавита ∑ понимается поиск конечной последовательности слов $\gamma_0, \gamma_1, \ldots, \gamma_n, \ldots$, где $\gamma = \gamma_0$ и для каждого $i \geq 1$ слово γ_i получено из γ_{i-1} в результате применения первой сверху подходящей марковской подстановки из схемы S. Вычислдительный процесс, т.е. поиск слова γ_n, заканчивается в одном из двух случаев:

1) слово γ_n получено из γ_{n-1} в результате применения заключительной подстановки;

2) ни одна из подстановок схемы S неприменима для слова γ_n.

Последний член последовательности γ_n называется **результатом применения нормального алгоритма** к слову γ. Тогда говорят, что нормальный алгоритм перерабатывает γ в $\gamma_n = f(\gamma)$. Если в процессе вычисления последовательность будет бесконечной и последний член получить невозможно, то говорят, что данный **нормальный алгоритм неприменим** к слову γ.

Например, пусть дан алфавит ∑={a,b}. Рассмотрим схему нормального алгоритма S:

{ba → ab.

Данный алгоритм перерабатывает любое слово алфавита ∑={a,b} в слово, которое сначала начинается символами a и заканчивается символами b. Вычисление слова γ = abbbab запишем последовательностью: γ_0=abb**ba**b, γ_1=ab**ba**bb,

γ_2=a**ba**bbb, γ_3=aabbbb (жирным шрифтом выделен фрагмент слова для замены). В данном алгоритме алфавиты $\sum$ и $\sum'$ совпадают.

Примером неприменимого ни к одному слову алфавита $\sum$={a,b} является алгоритм со схемой:

$$\begin{cases} b \to b \\ a \to a. \end{cases}$$

2.3. Пусть дан алфавит $\sum$={a,b} и схема алгоритма:

$$\begin{cases} aa \to .b \\ a \to a. \end{cases}$$

Примените его к следующим словам:

a) abaab; c) ababa; e) aabba;
b) baabaab; d) bbbbb; f) babbab.

Решение. Легко проверить, что алгоритм применим для всех слов, содержащих хотя бы одно вхождение символов aa или для слов, вообще не содержащих символы a. Это слова п. п. a), b), d), e).

Алгоритм не применим для слов, для которых выполняется вторая подстановка схем: п. п. c) и f).

2.4. Нормальный алгоритм в алфавите $\sum$={a,b} задан схемой:

$$\begin{cases} ca \to aac \\ cb \to bc \\ c \to . \\ \to \end{cases}$$

Примените его к следующим словам:

a) abb; c) abbab; e) babaab;
b) baba; d) bbb; f) aaaba.

Выявите закономерность в работе алгоритма.

Решение. a) Расширенный алфавит $\sum'$ ={a,b,c} алгоритма содержит три символа. Здесь c – служебный символ, появляется в начале исходного слова (подстановка 4), выполняет свое назначение (подстановки 2 и 3) и уходит после выполнения третьей заключительной подстановки. При этом работа алгоритма будет завершена. Вычисление слова abb представим следующей последовательностью: γ_0=abb, γ_1=cabb, γ_2=aacbb, γ_3=aabcb, γ_4=aabbc, γ_5=aabb.

2.5. Применим ли НАМ, заданный схемой:

$$\begin{cases} ca \to aac \\ cb \to bac \\ c \to ad \\ aaad \to . \\ bd \to bd \\ ad \to ad \\ db \to ca \\ \to c \end{cases}$$

к следующим словам:

a) aa; c) aba; e) abbba;
b) bbb; d) bababa; f) bbabb?

2.6. Постройте НАМ, который преобразует каждое слово в алфавите $\sum$={a,b} в:

a) пустое слово;

b) слово ab;

c) слово, к которому приписано справа слово β в этом же алфавите;

d) слово с удвоенными символами a и b.

e) слово, буквы которого записаны в обратном порядке;

f) удвоенное слово.

Решение. a) Для получения пустого слова достаточно удалить все символы исходного слова. Это можно реализовать следующей схемой нормального алгоритма:

$$\begin{cases} a \to \\ b \to \end{cases}$$

Здесь вместо пустого слова e мы ничего не пишем.

c) Построим нормальный алгоритм, который к любому слову α алфавита $\sum$={a,b} приписывает справа слово β в этом же алфавите, т.е. слово α перерабатывает в слово αβ. Схема нормального алгоритма примет вид:

$$\begin{cases} ca \to ac \\ cb \to bc \\ c \to .\beta \\ \to c \end{cases}$$

Вспомагательный символ с расширенного алфавита $\sum'$ ={a,b,c} в последовательности вычисления появляется благодаря последней подстановке и, пробегая все символы исходного слова α, исчезает в результате замены на слово β. Например, вычисление слова α=aba при β=bb можно представить следующей последовательностью: γ_0=aba, γ_1=caba, γ_2=acba, γ_3=abca, γ_4=abac, γ_5=ababb. Обратите внимание на порядок марковских подстановок в схеме алгоритма. Подстановка → c занимает именно последнюю строку, так как

должна срабатывать один раз, иначе произойдет зацикливание программы (бесконечное добавление символа с в начале слова), и нормальный алгоритм будет неприменим.

Нормально вычислимые функции. Функция f называется **нормально вычислимой**, если найдется такой нормальный алгоритм Маркова, который каждое слово γ алфавита ∑ из области определения функции перерабатывает в слово f(γ).

Однако НАМ позволяет производить вычисления не только словарных функций, но и числовых. Для этого любое неотрицательное число **x** будем изображать словом в алфавите ∑={1} из x+1 единицы. Например, число x=0 будет представлено словом 1, а x=3 словом 1111. Таким образом, для вычисления числовой функции f(x) необходимо построить такой НАМ, который будет перерабатывать слово из x+1 единицы в слово с числом единиц равных f(x)+1, т.е.

$$f: \underbrace{11\ldots1}_{x+1} \rightarrow \underbrace{11\ldots1}_{f(x)+1}$$

Например, схема нормального алгоритма {1 → вычисляет числовую функцию f(x)=0, а схема { → .1 вычисляет f(x)=x+1. Где пустой символ **e** может не отображаться в алгоритме.

Если требуется вычислить функцию от нескольких переменных, то в начальном слове используют символы для разделения аргументов (∗ ,°, +, - и т. п.), в процессе вычисления эти символы должны уйти.

Числовая функция $f(x_1,x_2,\ldots,x_n)$ является нормально вычислимой, если найдется НАМ, который будет применим к словам из области определения функции и неприменим к словам из других областей.

$$f: \underbrace{11\ldots1}_{x_1+1} * \underbrace{11\ldots1}_{x_2+1} *\ldots* \underbrace{11\ldots1}_{x_n+1} \rightarrow \underbrace{11\ldots1}_{f(x_1,x_2,\ldots,x_n)+1}$$

Здесь символ * используется как разделитель аргументов.

Например, вычисление функции f(x,y)=x+y можно представить следующей схемой нормального алгоритма

{ +1 → .

В качестве разделителя переменных используем знак +. Пусть x=2 и y=3. Тогда процесс вычисление представим последовательностью: γ_0=111+**1**1111, γ_1=111111. Заключительное слово 111111 является изображением числа x+y=5.

Далее рассмотрим задания на конструирование нормальных алгоритмов Маркова для ряда основных числовых функций.

2.7. Построить нормальный алгоритм Маркова, который вычисляет числовые функции на множестве N_0:

a) f(x)=x+2; b) f(x)=2x; c) f(x)=x/2;

d) f(x)=x $\dot{-}$1=$\begin{cases} x-1, & x>0 \\ 0, & x=0 \end{cases}$ (усеченная разность);

e) f(x)=[x/2] (x div 2, целая часть от деления x на 2);

f) f(x)=x mod 2 (остаток от деления x на 2);

g) f(x)=sgn(x)= $\begin{cases} 0, & x=0 \\ 1, & x>0 \end{cases}$ (знак числа);

h) f(x,y)= x $\dot{-}$ y; k) f(x,y)=min(x,y);

i) f(x,y)=|x-y|; l) f(x,y)=max(x,y);

j) f(x,y)=x-y; m) f(x,y)=x·y;

n) f(x,y)=|x-y| mod 2; o) f(x,y)=(x div 2)+(y mod 2).

Решение. b) Сконструируем нормальный алгоритм удвоения числа x. Вычисление функции f(x)=2x представим следующей схемой НАМ:

$$\begin{cases} a1 \to 11a \\ 1a \to . \\ \to a. \end{cases}$$

Последняя подстановка вводит вспомагательный символ a, который затем удваивает единицы числа x и исчезает, оставляя результат вычисления. Работа этого НАМ для числа x=3 состоит из последовательности слов: 1111, **a1**111, 11**a1**11, 1111a**1**1, 111111a**1**, 11111111a, 1111111. Слово 1111111 является изображением числа 6.

c) Рассмотрим нормальный алгоритм, вычисляющий функцию f(x)=x/2. Эта функция определена только на множестве четных чисел. Поэтому построим такой НАМ, который выдает результат для четного аргумента и будет работать вечно для нечетного, где алгоритм должен быть неприменим. Схему этого алгоритма запишем так:

$$\begin{cases} a11 \to 1a \\ a1 \to .1 \\ a \to a \\ \to a. \end{cases}$$

Подстановка a1→.1 срабатывает для четного x и после выполнения выдает ответ, если аргумент нечетный, вычисление зациклится на третьей подстановке. Функция вида f(x)=x/2 называют частично вычислимыми. Примерами таких функций являются f(x)=x-1, f(x)= $\sqrt{x}$ или f(x,y)=x-y.

h) Схема алгоритма вычисления функции f(x,y)=x $\dot{-}$ y:

$$\begin{cases} 1-1 \to - \\ -1 \to - \\ - \to .1. \end{cases}$$

Опишем работу этого алгоритма для x=4 и y=2 последовательностью слов: 11111 – 111, 1111 – 11, 111 – 1, 11 – , 111. Слову 111 соответствует результат вычисления 4 — 2 = 2.

Принцип нормализации Маркова. Разработчик теории нормальных алгоритмов математик А.А. Марков выдвинул гипотезу – **принцип нормализации**, согласно которому любой алгоритм, в частности и вычисление числовой функции, может быть реализован некоторым нормальным алгоритмом Маркова. Гипотеза выдвинута на основании математического и практического опыта, поэтому не носит строгий характер и не может быть доказана.

3. МАШИНЫ ТЬЮРИНГА

В 1937 г. английский математик Алан Тьюринг для уточнения понятия алгоритма предложил математическую модель вычислительной машины, которую в последствии назвали его именем. Машина Тьюринга есть такой же математический объект, как пространство, функция, производная и т. д. Она отражает объективную реальность, моделирует вычислительную деятельность человека.

Машина Тьюринга состоит из бесконечной в обе стороны ленты, разбитой на ячейки, и автомата (головки), которая управляется программой.

В каждой ячейке может быть записан только один символ некоторого конечного алфавита. Головка машины на каждом шаге работы указывает на одну ячейку и находится в одном из конечных состояний. Согласно программе, считывая символ в ячейке, обозреваемой головкой, и находясь в некотором состоянии, машина Тьюринга может менять свое состояние и при этом либо записывать в ячейку другой символ алфавита, либо сдвигаться влево или вправо по ленте на одну ячейку.

Опишем машину Тьюринга более подробно.

Внешний алфавит. Конечное множество символов $\sum=\{a_0,a_1,\ldots,a_n\}$, записанных на ленте. Одна из букв этого алфавита должна представлять собой пустой символ, именно она записана в пустую ячейку ленты. Условимся, что этой буквой является a_0 (или просто 0).

Внутренний алфавит. В каждый момент времени машина Тьюринга может находиться в одном из внутренних состояний конечного множества $Q=\{q_0,q_1,\ldots,q_m\}$. Среди состояний выделяют два – начальное q_1 и конечное (заключительное) q_0. Машина Тьюринга начинает работу в состоянии q_1. Попав в состояние q_0, машина завершает работу.

Программа. Работа машины Тьюринга определяется программой, которая состоит из команд трех типов:

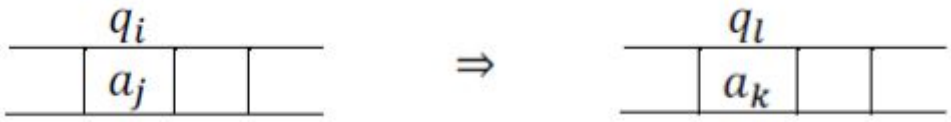
I. $q_i a_j a_k q_l$

II. $q_i a_j R q_l$

III. $q_i a_j L q_l$

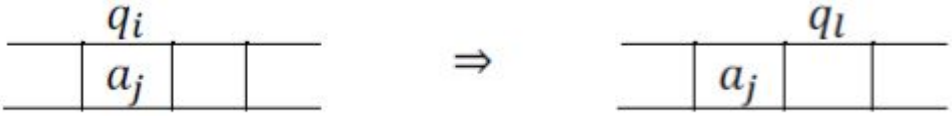
где $1 \le i \le m$; $0 \le l \le m$; $1 \le j, k \le n$; R – от английского right (право); L – от английского left (лево). Команда первого типа означает, что головка машины Тьюринга, находясь в состоянии q_i, и обозревая ячейку с символом a_j, осу-

ществляет замену на символ a_k и переходит в состояние q_l. Действие этой команды продемонстрировано на рисунке:



Согласно второй и третьей командам при тех же условиях головка осуществляет сдвиг на одну ячейку вправо или влево соответственно и переходит в новое состояние q_l. Это действие можно представить графически:

для команды второго типа (сдвиг головки вправо)



для команды третьего типа (сдвиг головки влево)

Выполнение одной команды машины Тьюринга называется **шагом.** Программа состоит из конечного числа команд. Так как работа машины Тьюринга полностью определяется ее состоянием q_i в данный момент и содержимым ячейки a_j, то для каждых q_i и a_j программа должна совершать только одну команду, начинающуюся парой q_ia_j. Такая пара называется **ситуацией.**

Другими словами, среди команд программы нет никаких двух, содержащих одинаковые ситуации.

Программу машины Тьюринга удобно представлять в виде таблицы или ориентированного графа.

Теперь дадим строгое формализованное определение машины Тьюринга. **Машиной Тьюринга** (далее МТ) называется структура вида

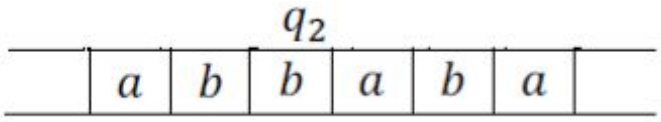
$$MT = <\sum, Q, a_0, q_0, q_1, P> \qquad (4)$$

где $\sum$ – внешний алфавит, Q – алфавит внутренних состояний и P – программа машины Тьюринга. Символ a_0 и состояния q_0, q_1 в структуре (4) говорят о том, что они обязательно присутствуют в любой МТ.

Под **конфигурацией** МТ будем понимать любое слово в множестве $\sum$ x Q, удовлетворяющее условиям:

- оно не содержит более одного состояния из Q;
- состояние в этом выражении не должно занимать крайнюю правую позицию.

Например, пусть на ленте МТ с внешним алфавитом $\sum = \{a_0,a,b\}$ и состояниями $Q=\{q_0,q_1,q_2,q_3\}$ записано слово abbaba. Головка МТ, находясь в некоторый момент времени в состоянии q2, обозревает третий символ. Графически это можно представить так:



Этой картине соответствует конфигурация ab**q_2**baba, которая указывает на все символы, записанные в данный момент на ленте с указанием состояния МТ и того символа, который обозревает головка. Таким образом, конфигурации однозначно изображают ленту МТ с информацией в текущий момент.

Каждый шаг выполнения одной из команд программы машины Тьюринга сопровождается переходом от одной конфигурации к другой. Конфигурация α переходит в конфигурацию β в одном из трех случаев:

1. $\exists$ w, v $\in\sum^*$, что $\alpha = wq_ia_jv$ и $\beta = wq_la_kv$. Переход осуществляется по команде первого типа;

2. $\exists$ w, v $\in\sum^*$, что $\alpha = wq_ia_jv$ и $\beta = wa_jq_lv$. (выполнена команда второго типа $q_ia_jRq_l$);

3. $\exists$ w, v $\in\sum^*$, что $\alpha = wa_kq_ia_jv$ и $\beta = wq_la_ka_jv$. (выполнена команда третьего типа $q_ia_jLq_l$);

Переход конфигурации α к конфигурации β принято обозначать так $\alpha\rightarrow\beta$.

Замечание. Если $\alpha \rightarrow \beta$ и $\alpha \rightarrow \gamma$, то $\beta = \gamma$. Что указывает на детерминированность алгоритма.

Конфигурация α называется **заключительной,** если не существует такого β, что выполняется переход $\alpha \rightarrow \beta$. Заключительная конфигурация содержит конечное состояние q_0.

Машина Тьюринга, как и нормальный алгоритм Маркова, служит для переработки слов некоторого алфавита ∑. Перерабатываемые слова в МТ записаны последовательно в ячейках ленты. В результате выполнения команд программы содержимое ячеек изменяться, таким образом, слова перерабатываются.

Под **вычислением** МТ понимается поиск такой конечной последовательности конфигураций $\alpha_0,\alpha_1,\ldots,\alpha_n$, что для каждого $i\in\{0,1,\ldots,n-1\}$ выполняется переход $\alpha_i \rightarrow \alpha_{i+1}$, и конфигурация α_n является заключительной.

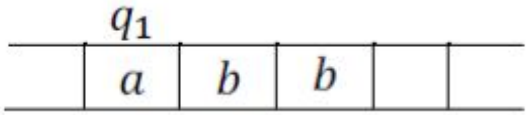
Вычисление МТ начинается и заканчивается с так называемого **стандартного положения**, когда слово записано в ячейках ленты и головка обозревает крайнюю левую ячейку слова. **Слово w перерабатывается МТ в слово v**, если от начальной конфигурации q_1w, воспринимаемой в стандартном положении, машина после выполнения конечного числа команд приходит к заключительной конфигурации q_0v. Такой переход принято обозначать $q_1w \mapsto q_0v$.

Например, пусть дана МТ с внешним алфавитом $\sum=\{a_0,a,b\}$, состояниями $Q = \{q_0,q_1,q_2\}$ и программой: q_1aRq_1, q_1bRq_1, $q_1a_0aq_2$, q_2aLq_2, q_2bLq_2, $q_2a_0Rq_0$. Программу МТ удобно представить в виде таблицы:

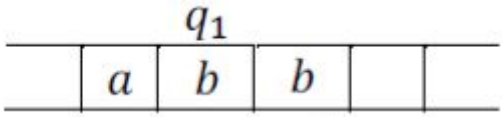
Σ \ Q	q_1	q_2
a_0	aq_2	Rq_0
a	Rq_1	Lq_2
b	Rq_1	Lq_2

Посмотрим, в какое слово переработает машина Тьюринга слово abb из стандартного положения. Выпишем последовательно конфигурации каждого шага вычисления.

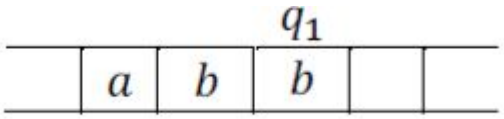
Имеем стандартное начальное положение:

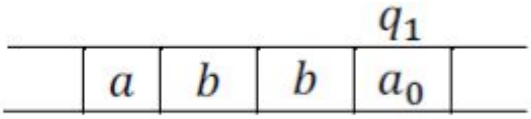


Ему соответствует начальная конфигурация q_1abb. На первом шаге действует команда q_1aRq_1 (ситуация q_1a), в результате которой выполняется переход к следующей конфигурации aq_1bb:

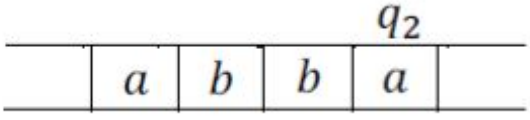


На втором и третьем шаге действует команда q_1bRq_1 и на машине создаются конфигурации abq_1b и $abbq_1a_0$:

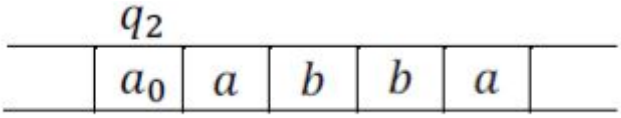




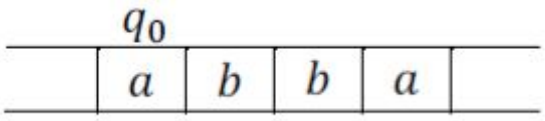
На четвертом шаге на месте пустой ячейки (символ a_0) появляется символ a, обусловленный командой $q_1a_0aq_2$. Имеем конфигурацию $abbq_2a$:



Далее командами q_2aLq_2 и q_2bLq_2 головка машины сдвигается влево до пустой ячейки. Выполняются следующие переходы конфигураций:
$abq_2ba \rightarrow aq_2bba \rightarrow q_2abba \rightarrow q_2a_0abba$. Последняя конфигурация этого сдвига:



Наконец, последний шаг, обусловленный командой $q_2a_0Rq_0$, приводит к заключительной конфигурации в стандартном положении.



Таким образом, слово abb переработано в слово abba, т.е. $q_1abb \mapsto q_0abba$.

Приведем последовательность конфигураций при переработке этой машиной из начального положения слова baab. Кратко переходы по конфигурациям можно представить так:
$q_1baab \rightarrow bq_1aab \rightarrow baq_1ab \rightarrow baaq_1b \rightarrow baabq_1a_0 \rightarrow baabq_2a \rightarrow baaq_2ba \rightarrow baq_2aba \rightarrow bq_2aaba \rightarrow q_2baaba \rightarrow q_2a_0baaba \rightarrow q_1baaba$. Слово baab переработано машиной в слово baaba.

Нетрудно заметить, что данная МТ приписывает к данному слову справа символ a. Можно сказать, что машина Тьюринга вычисляет словарную функцию $f(w)=wa$, где $w \in \sum^*$.

3.1. Имеется машина Тьюринга с внешним алфавитом $\sum=\{a_0,a,b\}$, алфавитом внутренних состояний $Q = \{q_0,q_1,q_2\}$ и программой: $q_1aa_0q_2$, $q_1ba_0q_3$, $q_2a_0Rq_4$, $q_3a_0Rq_5$, q_4aRq_4, q_4bRq_4, q_5aRq_5, q_5bRq_5, $q_4a_0aq_0$, $q_5a_0bq_0$.

1) Запишите программу МТ в виде таблицы;

2) Определите, в какое слово перерабатывает МТ каждое из следующих слов из стандартного положения:

a) abab; b) abbba; c) bbabaa; d) baabbbab; e) aabbabaaba.

Изобразите схематически переходы между конфигурациями;

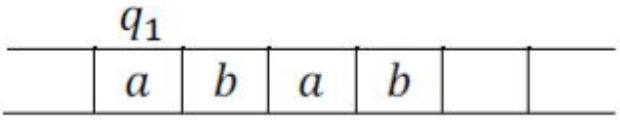
3) Понаблюдайте за результатами перерабатываемых слов и определите, какое действие реализует МТ над словами алфавита {a,b};

4) Доработайте программу так, чтобы заключительная конфигурация имела стандартное положение.

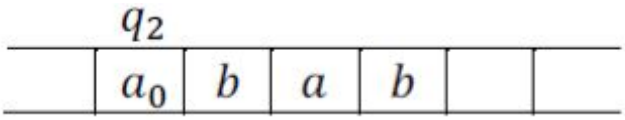
Решение. Программу МТ запишем в виде таблицы:

$\sum$ \ Q	q_1	q_2	q_3	q_4	q_5
a_0		Rq_4	Rq_5	aq_0	bq_0
a	a_0q_2			Rq_4	Rq_5
b	a_0q_3			Rq_4	Rq_5

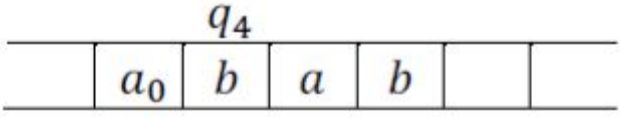
a) Слово abab на ленте в стандартном положении:



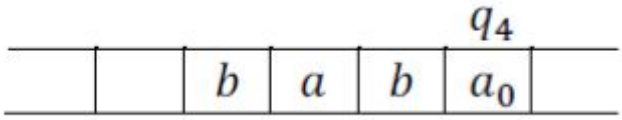
Начальная конфигурация примет вид q_1abab. Согласно команде $q_1aa_0q_2$ машина стирает первый символ и переходит в новое состояние q_2:



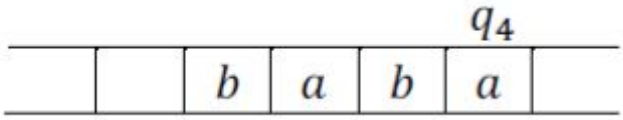
На втором шаге согласно команде $q_2a_0Rq_4$ головка сдвигается вправо и переходит в состояние q_4:



По командам q_4aRq_4 и q_4bRq_4 машина Тьюринга в состоянии q_4 двигается вправо по символам слова до первой пустой ячейки:



Переход по конфигурациям запишем кратко: $q_4bab \rightarrow bq_4ab \rightarrow baq_4b \rightarrow babq_4a_0$. На последнем шаге по команде $q_4a_0aq_0$ машина запишет в пустую ячейку символ a и переходит в заключительное состояние q_0:



Таким образом, слово abab перерабатывается машиной в слово baba. Заметим, что заключительная конфигурация $babq_4a$ не имеет стандартное положение.

3.2. Дана машина Тьюринга с внешним алфавитом $\sum=\{a_0,a\}$, внутренними состояниями $Q=\{q_0,q_1,q_2\}$ и программой: $q_1aa_0q_2$, $q_2a_0Rq_1$, $q_1a_0a_0q_0$. Запишите программу в виде таблицы. Определите, в какое слово перерабатывает МТ каждое из следующих слов из стандартного положения:

a) aaa; b) aaa_0a; c) aa_0aaaa; d) $aaaaa_0a$.

Запишите переходы между конфигурациями.

3.3. Машина Тьюринга задана программой, записанной в виде таблицы:

Σ \ Q	q_1	q_2	q_3
a_0	Lq_3		Rq_0
a	bq_2	Rq_1	Lq_3
b	aq_2	Rq_1	Lq_3

а) Запишите программу в виде последовательности команд;

б) Определите, в какое слово перерабатывает машины каждое из следующих слов из начального положения: abba; bbaa; aabaab; aaaa; bbbaaa; abababab.

с) Понаблюдайте за результатами перерабатываемых слов и определите, какое действие реализует МТ над словами алфавита {a,b}.

3.4. Дана машина Тьюринга с внешним алфавитом ∑={0,1} (0 – символ пустой ячейки, 1 – информативный символ), внутренними состояниями Q={q_0,q_1,q_2,q_3,q_4} и прогрммой: $q_1$1Lq_2, $q_2$01q_3, $q_3$1Lq_4, $q_4$01q_0. Запишите программу в виде таблицы. Определите, в какое слово перерабатывает МТ каждое из следующих слов из стандартного положения:

а) 11; b) 111; c) 1111; d) $\underbrace{11\ldots1}_{n}$.

Находится ли заключительная конфигурация в стандартном положении? Какое действие реализует МТ над единицами, записанными на ленте? Можно ли сконструировать МТ с меньшим числом состояний и реализующую это действие?

3.5. Машина Тьюринга задана программой, записанной в виде таблицы:

Σ \ Q	q_1	q_2	q_3
0	Rq_1	Rq_2	Rq_0
1	$0q_2$	$0q_3$	

Запишите программу в виде последовательности команд. Определите, в какое слово перерабатывает МТ каждое из следующих слов из стандартного положения:

а) 111; b) 11; c) 1; d) $\underbrace{11\ldots1}$, n>2.

Проследите за общей закономерностью в работе машины. Для каждого ли перерабатываемого слова МТ остановит свою работу? Какое действие реализует МТ над единицами, записанными на ленте?

3.6. Машина Тьюринга с внешним алфавитом ∑={0,1,*} (0 – символ пустой ячейки, * – разделитель информативных единичек) задана программой:

∑ \ Q	q_1	q_2	q_3
0		Rq_3	Rq_0
1	Rq_1	Lq_2	$0q_3$
*	$1q_2$		

Определите, в какое слово перерабатывает МТ каждое из следующих слов из стандартного положения:

a) 11*111; b) 111*1; c) *111; d) $\underbrace{11\ldots1}_{n}*\underbrace{11\ldots1}_{m}$, n, m∈N.

Проследите за общей закономерностью в работе машины. Какое действие реализует МТ над единицами, записанными на ленте?

Решение. a) Переработку слова 11*111 из начального положения запишем последовательностью конфигураций:
$q_1$11*111→ 1$q_1$1*111 → 11q_1*111 → 11$q_2$1111→ 1$q_2$11111→ $q_2$111111→ $q_2$0111111 → $q_3$111111 → $q_3$011111 → $q_0$11111. Таким образом, слово 11*111 перерабатывается машиной в слово 11111.

3.7. Пусть машина Тьюринга задана программой:

∑ \ Q	q_1	q_2	q_3	q_4	q_5
0		Lq_3		Lq_5	Rq_0
1	Rq_1	Rq_2	$0q_1$		Lq_5
*	Rq_2		$0q_4$		

Определите, в какое слово перерабатывает МТ каждое из следующих слов из стандартного положения:

a) 111*111; b) 11*1111; c) *111; d) d) $\underbrace{11\ldots1}_{n}*\underbrace{11\ldots1}_{m}$, n, m∈N.

Проследите за общей закономерностью в работе машины. Какое действие реализует МТ над единицами, записанными на ленте?

3.8. Постройте машину Тьюринга, которая удаляет на ленте слова в алфавите {a,b}.

Решение. В качестве внешнего алфавита возьмем множество ∑={a_0,a,b}, где a_0 – символ пустой ячейки.

Пусть на ленте записано слово в алфавите {a,b}, и головка МТ обозревает первый символ этого слова, т.е. машина находится в стандартном положении.

Начнем с того, что сотрем первый символ слова на ленте и перейдем к обозрению следующей правой ячейке. На каждом таком переходе процесс удаления должен повторяться.

Команды, осуществляющие эти действия: $q_1aa_0q_2$; $q_1ba_0q_2$; $q_2a_0Rq_1$. В состоянии q_1 машина стирает символы. Попадая в состояние q_2, действие по удалению зацикливается переходом в состояние q_1.

Если в состоянии q_1 головка обозревает пустую ячейку, то это говорит о том, что результат получен, и машина должна перейти в заключительное состояние командой $q_1a_0a_0q_0$.

Запишем составленную программу в виде таблицы:

$\sum$ \ Q	q_1	q_2
a_0	a_0q_0	Rq_1
a	a_0q_2	
b	a_0q_2	

Заметим, что если на ленте записано слово, у которого между символами a, b стоит пустая ячейка, то стерты будут только символы до этой ячейки. Например, слово $abba_0ab$ перерабатывается машиной в слово ab.

3.9. Постройте машину Тьюринга, которая создает на ленте копию слова в алфавите {1}.

Решение. В качестве внешнего алфавита возьмем множество $\sum=\{0,1\}$, где 0 – символ пустой ячейки. Пусть на ленте записано слово из n единиц, где $n \in N$, и машина Тьюринга находится в стандартном положении.

Работа МТ будет состоять из n циклов. К началу i-го цикла ($i \le n$) конфигурация будет такова:

$$\underbrace{11\ldots1}_{i-1}\ q_1\ \underbrace{11\ldots1}_{n-i+1}\ 0\ \underbrace{11\ldots1}_{i-1},$$

т.е. продублировано уже i-1 единица. Ячейка с пустым символом 0 является разделителем копии от оригинала.

Очередной цикл будет проходить в три этапа: I – запоминание i-й единицы; II – поиск крайней пустой клетки, в которую записываются копия; III – возвращение головки к стертому символу и его восстановление, переход к копированию i+1-го символа.

Первый этап осуществляется командой q_110q_2, т.е. запоминание достигается стиранием 1-й единицы.

Поиск крайней пустой ячейки и запись копии достигается следующими командами: q_20Rq_3, q_31Rq_3, q_30Rq_4, q_41Rq_4 (сдвиг головки вправо), q_401q_5 (запись копии).

На третьем этапе головка движется обратно к удаленному оригиналу и восстанавливает его: $q_5 1 L q_5$, $q_5 0 L q_6$, $q_6 1 L q_6$ (сдвиг головки влево), $q_6 0 1 q_7$ (восстановление оригинала). Цикл замыкается командой $q_7 1 R q_1$.

Наконец, после копирования всех единиц оригинала возникает ситуация $q_1 0$, теперь головку МТ необходимо вернуть в стандартное положение и завершить работу машины. Соответствующие команды таковы: $q_1 0 L q_8$, $q_8 1 L q_8$, $q_8 0 R q_0$.

Запишем составленную программу:

Σ \ Q	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
0	Lq_8	Rq_3	Rq_4	$1q_5$	Lq_6	$1q_7$		$0Rq_0$
1	$0q_2$		Rq_3	Rq_4	Lq_5	Lq_6	Rq_1	Lq_8

3.10. Постройте машину Тьюринга с внешним алфавитом $\{a_0,a,b\}$, которая из стандартного положения переставляет последний символ слова в алфавите $\{a,b\}$ в начало.

3.11. Постройте машину Тьюринга с внешним алфавитом $\{a_0,a,b\}$, которая каждое слово в алфавите $\{a,b\}$ перерабатывает в слово, символы которого записаны в обратном порядке, исходя из стандартного положения.

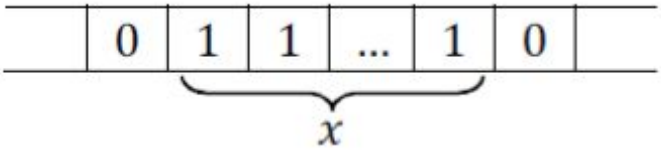
Указание. Символы симметрично переносятся влево от исходного слова. Например, $abaab \to aa_0baab \to baa_0a_0aab \to abaa_0a_0a_0ab \to aabaa_0a_0a_0a_0b \to baaba$.

3.12. Постройте машину Тьюринга с внешним алфавитом $\{0,1\}$, которая каждое слово длиной n в алфавите $\{1\}$ перерабатывает из стандартного положения: a) в пустое слово; b) в слово длиной n+1 (по аналогии с заданием 4); c) в слово длиной n-1, если $n \geq 1$, или в слово 0, если n=0 (по аналогии с заданием 5); d) в 1, если n – нечетно, или в 0, если n – четно.

3.13. На ленте записаны два конечных набора из n и m единиц, разделенных звездочкой. Составьте программу МТ, которая из стандартного положения: a) стирает все единицы, расположенные слева от звездочки (n единиц); b) стирает слева столько единиц, сколько их было справа, все единицы справа тоже должны быть стерты (в результате должно остаться n-m единиц, $n \geq m$); c) стирает набор из меньшего числа единиц.

Числовая функция называется вычислимой по Тьюрингу, если существует машина Тьюринга, которая вычисляет ее значения там, где она определена, и работает вечно на тех значениях аргументов, где функция не определена.

Условимся натуральное число x изображать на ленте единицами:



Для натурального x введем обозначение: $\underbrace{11\ldots1}_{x}=1^x$.

Значения нескольких аргументов $x_1,x_2,\ldots,x_n$ на ленте МТ будем изображать в алфавите {0,1} в виде следующего слова:

$$0\underbrace{11\ldots1}_{x_1}0\ \underbrace{11\ldots1}_{x_2}0\ldots0\underbrace{11\ldots1}_{x_n}0,$$

или кратко $01^{x1}01^{x2}0\ldots01^{xn}0$.

Говорят, что машина Тьюринга **правильно вычисляет** числовую функцию $f(x_1,x_2,\ldots,x_n)$ на тех наборах значений аргументов, для которых функция определена, если

$$q_101^{x1}01^{x2}0\ldots01^{xn}0\mapsto q_001^{f(x1,x2,\ldots,xn)}0. \qquad (5)$$

Если функция не определена на некотором наборе значений аргументов, то машина Тьюринга должна работать бесконечно.

Заметим, что стандартным начальным положением будет являться то положение, при котором в состоянии q_1 головка МТ обозревает пустую ячейку перед первой единицей записанного слова, ситуация q_10 будет всегда являться начальной. Это верно и для стандартного конечного положения.

3.14. Постройте машины Тьюринга, которые правильно вычисляют следующие функции.

a) f(x)=0; b) f(x)=x+1; c) f(x)=x $\dot{-}$ 1; d) f(x)=x mod 2; e) f(x)=2x;

f) f(x,y)=x; g) f(x,y)= y; h) f(x,y)=x+y; i) f(x,y)=x-y, x≥y;

j) f(x,y)=max(x,y).

Указание. Для пунктов a)-d) см. задание 12; для п. e) – задание 9; для п. f) – задание 7; для п. h) – задание 6; для п. п. g), i) и j) см. задание 13.

Решение. a) Согласно (5) функция f(x)=0 осуществляет переход конфигураций: $q_101^x0\mapsto q_00$. Ее программа: q_10Rq_2, q_210q_1, q_200q_0.

d) Функция f(x) = x mod 2, позволяющая найти остаток от деления натурального x на 2, осуществляет переход конфигураций: $q_101^x0\mapsto q_001^{x \bmod 2}0$. Если x четное, то заключительной конфигурацией должна быть q_00, для нечетного x – q_0010. Программу напишем по аналогии с заданием 12d): q_10Rq_2, q_210q_3, q_30Rq_4, q_410q_1, q_200q_0, q_401q_5, q_51Lq_0.

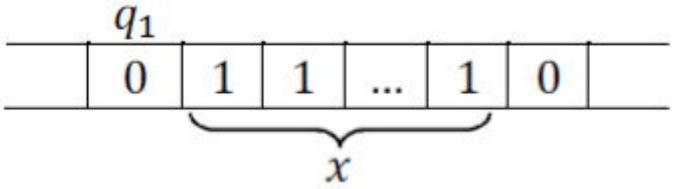
e) Напишем программу, которая для вычисления функции f(x)=2x, переходит из стандартного начального положения q_101^x0 в заключительное $q_001^{2x}0$. Для этого сначала создадим копию слова 1^x (см. задание 9), в результате на ленте будет записано слово 01^x01^x0. Затем выполним переход к заключительному стандартному положению $q_001^{2x}0$. Исправим и дополним программу из задания 9, в итоге получим: q_10Rq_2, q_210q_3, q_30Rq_4, q_41Rq_4, q_40Rq_5, q_51Rq_5, q_501q_6, q_61Lq_6, q_60Lq_7, q_71Lq_7, q_701q_8, q_81Rq_2, (запись копии завершена), q_201q_8, q_81Lq_8, q_80Rq_9, q_910q_0.

h) Машина Тьюринга правильно вычисляет функцию f(x,y)= x+y, если $q_101^x01^y0\mapsto q_001^{x+y}0$. Вместо разделителя 0 между числами x и y следует за-

писать символ 1, затем удалить первую единицу исходного слова и остановить работу машины. Ее программа примет вид: $q_1 0 R q_2$, $q_2 1 R q_2$, $q_2 01 q_3$ (замена разделителя), $q_3 1 L q_3$, $q_3 0 R q_4$, $q_4 10 q_0$ (стираем первую единицу и останавливаем работу МТ).

3.15. Постройте машину Тьюринга, правильно вычисляющую функцию $f(x)=x/2$.

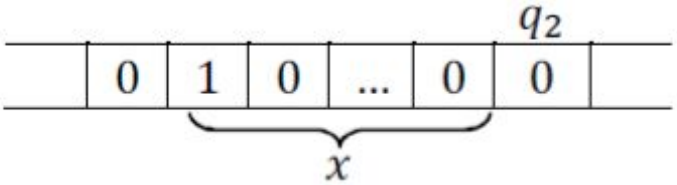
Решение. Функция $f(x)=x/2$ определена только на четном аргументе. Поэтому требуется написать программу, которая для четного числа x осуществляла бы переход в заключительное положение $q_0 01^{x/2} 0$, а для нечетного – работала бесконечно. Работа МТ начинается из начального стандартного положения $q_1 01^x 0$:



Пусть машина обозревает ячейки, двигаясь слева направо, и каждую вторую единицу переводит в ноль. Команды, осуществляющие эти действия: $q_1 0 R q_2$, $q_2 1 R q_3$, $q_3 10 q_1$.

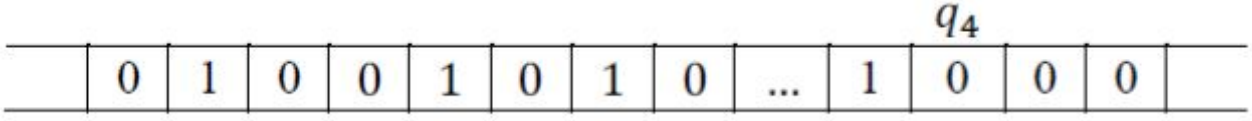
Если число x нечетно, возникает ситуация $q_3 0$. Машина продолжит движение неограниченно вправо командой $q_3 0 R q_3$. Работа машины зациклиться, как этого требует условие.

Если число x четно (ситуация $q_2 0$), то в результате выполнения команд имеем на ленте число единиц равное x/2:

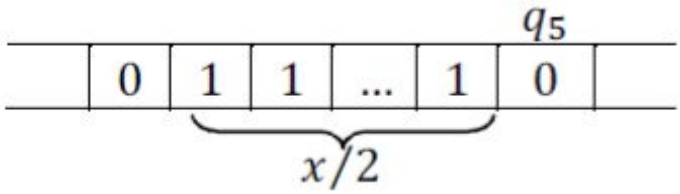


Остается сдвинуть единицы так, чтобы между ними не стояли нули. Для этого напишем цикл, который все единицы на ленте будет переносить влево от полученного слова через два разделителя 00. Один шаг цикла представляет собой удаление крайней правой единицы и ее восстановление слева.

Опишем команды тела цикла. Двигаемся по ленте влево до первой единицы: $q_2 0 L q_4$, $q_4 0 L q_5$. Стираем единицу и двигаемся по ячейкам из чередующихся единиц и нулей влево для восстановления единицы: $q_5 10 q_6$ (стираем), $q_6 0 L q_7$, $q_7 0 L q_8$, $q_8 1 L q_7$, $q_8 0 L q_9$, $q_9 1 L q_9$, $q_9 01 q_{10}$, (восстанавливаем), $q_{10} 1 R q_{10}$, $q_{10} 0 R q_{11}$, $q_{11} 0 R q_{12}$, $q_{12} 1 R q_{11}$, $q_{12} 0 L q_4$ (цикл замкнулся). После первого шага цикла на ленте будут записаны x/2 единиц:



Работа цикла будет завершена, когда все единицы слева будут перенесены в правую часть, ситуация $q_5 0$ укажет на конец перемещения. В результате на ленте будет записано слово:



Остается переместить головку в стандартное положение и завершить работу машины. Это делается с помощью команд: $q_5 0 L q_{13}$, $q_{13} 1 L q_{13}$, $q_{13} 00 q_0$.

Запишем программу МТ в табличной форме:

Σ \ Q	q_1	q_2	q_3	q_4	q_5	q_6	q_7
0	Rq_2	Lq_4	Rq_3	Lq_5	Lq_{13}	Lq_7	Lq_8
1		Rq_3	$0q_1$		$0q_6$		

Σ \ Q	q_8	q_9	q_{10}	q_{11}	q_{12}	q_{13}
0	Lq_9	$1q_{10}$	Rq_{11}	Rq_{12}	Lq_4	$0q_0$
1	Lq_7	Lq_9	Rq_{10}		Rq_{11}	Lq_{13}

Самостоятельно проследите за работой МТ, взяв в качестве исходных слов: 111, 1111, 111111.

3.16. Постройте машины Тьюринга, которые правильно вычисляют следующие функции: a) f(x)=x+2; b) f(x)=x $\dot{-}$ 2; c) f(x)=x mod 3; d) f(x)=3x;

e) f(x)=[x/2] (или x div 2, поиск частного от деления на 2); f) f(x)=sgn(x)= $\begin{cases} 0, \textit{если } x = 0 \\ 1 \textit{ если } x > 0 \end{cases}$ (знак числа); g) f(x,y)=x $\dot{-}$ y; h) f(x,y)=|x-y|; i) f(x,y)=x-y;

j) f(x,y)=min(x,y).

В настоящее время не представляется возможным придумать хоть какой-нибудь алгоритм, для которого нельзя было бы построить машину Тьюринга. В связи с этим математики принимают

Тезис Тьюринга. Любой алгоритм может быть реализован на машине Тьюринга.

Так называемая основная гипотеза теории алгоритмов, как и принцип нормализации Маркова, выдвинута на основании практического опыта, поэтому не может быть доказана.

Утверждение. Любая функция, вычислимая по Тьюрингу, будет являться нормально вычислимой, и наоборот.

4. РЕКУРСИВНЫЕ ФУНКЦИИ

В предыдущих двух параграфах были рассмотрены примеры функций, которые вычисляются на нормальных алгоритмах Маркова и машинах Тьюринга. В 1930-х гг. создание теории рекурсивных функций позволило описать все множество таких алгоритмически или эффективно вычислимых функций.

Примитивно рекурсивные функции. Далее будем рассматривать функции, заданные на множестве неотрицательных целых чисел $N_0=\{0,1,2,\ldots\}$ и принимающие значения на нем. При этом, эти функции могут быть как всюду определенными, так и частичными.

Назовем **исходными** (или базисными) следующие числовые функции:

1) O(x) = 0 (нуль-функция);

2) S(x) = x+1 (функция следования);

3) $I_n^m(x_1,x_2,...,x_n)=x_m$ (функции-проекции, $1 \le m \le n$).

Например, функциями-проекциями являются $I_1^1(x)=x$, $I_2^1(x,y)=x$, $I_2^2(x,y)=y$, $I_3^2(x_1,x_2,x_3)=x_2$ и др.

Вычислимость исходных функций на машинах Тьюринга и нормальных алгоритмах Маркова была установлена ранее.

Теперь введем два правила (или оператора) для получения новых из уже имеющихся исходных функций.

I. Оператор суперпозиции. Пусть даны числовые функции $h(x_1,x_2,\ldots,x_m)$ и $g_1(x_1,x_2,\ldots,x_n)$, $g_2(x_1,x_2,\ldots,x_n),\ldots$, $g_m(x_1,x_2,\ldots,x_n)$. Говорят, что n-местная функция f получена из m-местной h и n-местных g_1, $g_2,\ldots,g_m$ с помощью оператора суперпозиции, если справедливо равенство:

$f(x_1,x_2,\ldots,x_n)= h(g_1(x_1,x_2,\ldots,x_n),\ldots, g_m(x_1,x_2,\ldots,x_n))$.

Например, функция f(x)=1 получена из функций O(x) и S(x) в результате суперпозиции: S(O(x))=1. Функция f(x,y)=y+1 получена суперпозицией функций S(x) и $I_2^2(x,y): S(I_2^2(x,y))=y+1$.

II. Оператор примитивной рекурсии. Пусть даны числовые функции $h(x_1,x_2,\ldots,x_{n+1})$ и $g(x_1,x_2,\ldots,x_{n-1})$, $n \ge 0$. Говорят, что n-местная функция f получена из (n-1)-местной g и (n+1)-местной h с помощью **оператора примитивной рекурсии,** если справедливы равенства:

$$f(x_1,x_2,\ldots,x_{n-1},0)= g(x_1,x_2,\ldots,x_{n-1});$$

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

Здесь переменная x_{n+1} функции h есть функция f.

Случай n=1 рассмотрим отдельно. Одноместная функция f(x) получена из 0-местной g (константы) и 2-местной h с **помощью оператора примитивной рекурсии**, если: f(0)=g; f(x+1)=h(x,y), где y=f(x).

Например, функция f(x)=2x может быть получена с помощью примитивной рекурсии из константы g=0 и функции h(x,y) = y+2. Действительно f(0)=0; f(x+1)=2x+2=f(x)+2=y+2.

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

Так как исходные функции являются всюду определенными, и операторы суперпозиции и примитивной рекурсии сохраняют эту определенность, то из определения следует, что каждая примитивно рекурсивная функция является всюду определенной.

4.1. Докажите, что следующие функции примитивно рекурсивны:
a) f(x)=k, k>0; b) f(x)=x+k, k>0; c) f(x,y)=x+y; d) f(x)=kx, k>1; e) f(x,y)=x·y;
f) $f(x)=x^k$, k>1; g) $f(x)=a_kx^k+a_{k-1}x^{k-1}+\ldots+a_1x+a_0$, где k>0, $a_i\in N_0$, $i\in 0,1,\ldots,k$ (многочлены с натуральными коэффициентами).

Решение. a) Функция f(x)=k получается из исходных в результате применения только оператора суперпозиции. Для этого необходимо k-раз применить функцию следования S(x) к функции O(x). В самом деле, $f(x)=\underbrace{S(S(\ldots S}_{k}(O(x))\ldots))=k$.

c) Функция f(x,y)=x+y с помощью операторов суперпозиции и примитивной рекурсии может быть получена из исходных $I_1^1(x)$, $I_3^3(x,y,z)$ и S(x). Действительно, используя оператор примитивной рекурсии, получим:

f(x,0)=x=$I_1^1(x)$; f(x,y+1)=x+y+1=f(x,y)+1=z+1.

Функция h(x,y,z) получена из исходных при помощи оператора суперпозиции: h(x,y,z)=z+1=S($I_3^3(x,y,z)$).

e) Функция f(x,y)=x·y получается с помощью оператора примитивной рекурсии из одноместной функции O(x) и трехместной функции $I_3^1(x,y,z)+I_3^3(x,y,z)$:

x·0=0=O(x); x·(y+1)=xy+x=f(x,y)+x=z+x=$I_3^1(x,y,z)+I_3^3(x,y,z)$.

g) Примитивная рекурсивность многочлена с натуральными коэффициентами $f(x)=a_kx^k+a_{k-1}x^{k-1}+\ldots+a_1x+a_0$ следует из примитивной рекурсивности функций предыдущих пунктов a)-f).

Утверждение. Функция, полученная суперпозицией примитивно рекурсивных функций, сама примитивно рекурсивная. В частности, сумма и произведение примитивно рекурсивных функций есть функция примитивно рекурсивная. Условимся в дальнейшем использовать этот факт без подробного описания через определение примитивно рекурсивной функции.

4.2. Докажите, что следующие функции примитивно рекурсивны:

a) f(x)=x!; b) $f(x)=x\dot{-}1$; c) $f(x,y)=x\dot{-}y$; d) f(x,y)=|x-y|; e) f(x,y)=max(x,y);

f) f(x,y)=min(x,y);

g) $f(x)=sgn(x)=\begin{cases}0, \textit{если } x=0\\ 1 \textit{ если } x>0\end{cases}$; h) $f(x)=\overline{sgn}(x)=\begin{cases}0, \textit{если } x>0\\ 1 \textit{ если } x=0\end{cases}$.

Решение. b) Усеченная разность $f(x)=x\dot{-}1$ удовлетворяет следующим соотношениям: $g=f(0)=0\dot{-}1=O(x)$; $h(x,y)=f(x+1)=(x+1)\dot{-}1=x=I_2^1$, т.е. получена из исходных O(x) и I_2^1 с помощью оператора примитивной рекурсии.

g) Знак числа sgn(x) можно представить через исходные O(x) и S(x) с помощью операторов суперпозиции и примитивной рекурсии:

g=sgn(0)=O(x); h(x,y)=sgn(x+1)=1=S(O(x)).

Также эту функцию можно записать через усеченную разность, примитивная рекурсивность которой показана: $sgn(x)=1-\overline{sgn}(x)$.

Примитивная рекурсивность предикатов. Предикатом $P(x_1,x_2,\ldots,x_n)$ называется функция, аргументы которой принимают значения из некоторого множества (предметная область), а сама функция принимает логическое значение из множества {0,1}. Здесь 0 – ложь, 1 – истина.

Например, предикат P(x)=(x⋮2), заданный на множестве N. При четном аргументе предикат принимает истинное значение (P(2)=1), при нечетном – ложное (P(3)=0).

Двухместный предикат P(x,y)=(x>y), заданный на множестве Z, принимает ложное значение, если аргумент x не превосходит y, и истинное, наоборот. Так P(4,1)=1 (истина), при P(2,7)=0 (ложь).

Пусть n-местный предикат $P(x_1,x_2,\ldots,x_n)$ задан на множестве N_0.

Характеристической функцией этого предиката называется функция, заданная на том же множестве N_0, но принимающая числовые значения из множества {0,1}:

$$\chi_P(x_1,x_2,\ldots,x_n)=\begin{cases}0, \textit{если } P(x_1,x_2,\ldots,x_n) \textit{ ложно},\\ 1, \textit{если } P(x_1,x_2,\ldots,x_n) \textit{ истинно}.\end{cases}$$

Если характеристическая функция χ_P примитивно рекурсивна, то и предикат $P(x_1,x_2,\ldots,x_n)$ называется **примитивно рекурсивным**.

Утверждение. Если предикаты $P(x_1,x_2,\ldots,x_n)$ и $Q(x_1,x_2,\ldots,x_n)$ примитивно рекурсивны, то следующие предикаты примитивно рекурсивны:

$\overline{P}(x_1,x_2,\ldots,x_n)$, так как $\chi_{\overline{P}}=\overline{sgn}(\chi_P)$;

$P(x_1,x_2,\ldots,x_n)\wedge Q(x_1,x_2,\ldots,x_n)$, так как $\chi_{P\wedge Q}=\chi_P\cdot\chi_Q$;

$P(x_1,x_2,\ldots,x_n)\vee Q(x_1,x_2,\ldots,x_n)$, $\chi_{P\vee Q}=sgn(\chi_P+\chi_Q)$.

Утверждение. Из примитивной рекурсивности предиката $Q(x_1,x_2,\ldots,x_m)$ и функций $f_1(x_1,x_2,\ldots,x_n)$, $f_2(x_1,x_2,\ldots,x_n)$,…, $f_m(x_1,x_2,\ldots,x_n)$ следует примитивная рекурсивность предиката $P(x_1,x_2,\ldots,x_n)=Q(f_1(x_1,x_2,\ldots,x_n),\ldots, f_m(x_1,x_2,\ldots,x_n))$.

4.3. Докажите, что следующие предикаты примитивно рекурсивны:

a) P(x)=(x ⋮ 2); b) P(x,y)=(x=y); c) P(x,y)=(x≤y); d) P(x,y)=(x>y);

e) P(x,y,z)=(x+y=z); f) P(x,y,z)=(x·y=z).

Решение. a) Предикат $P(x)=(x \vdots 2)$ является примитивно рекурсивным, так как его характеристическую функцию $\chi_P(x)$ можно представить с помощью оператора примитивной рекурсии через примитивно рекурсивные функции: $\chi_P(0)=1$; $\chi_P(x+1)=\overline{sgn}(\chi_P(x))$.

d) Характеристическая функция предиката $P(x,y)=(x>y)$ является примитивно рекурсивной, так как ее можно записать в виде суперпозиции примитивно рекурсивных: $\chi_P(x,y)=sgn(x \dot{-} y)$. Значит предикат примитивно рекурсивен. Примитивная рекурсивность этого предиката может быть доказана через примитивную рекурсивность предиката п. c) этого задания, записанного как его отрицание. Согласно утверждению, $P(x,y)=\overline{(x>y)}=\overline{(x \le y)}$.

e) Примитивная рекурсивность предиката $P(x,y,z)=(x+y=z)$ следует из примитивной рекурсивности предиката $Q(x,y)=(x=y)$ и двух примитивно рекурсивных функций $f_1(x,y,z)=x+y$ и $f_2(x,y,z)=z$. Тогда согласно утверждению предикат $P(x,y,z)$ можно представить как суперпозицию предиката $Q(x,y)$ и функций $f_1(x,y,z)$, $f_2(x,y,z)$: $(x+y=z)=Q(f_1(x,y,z), f_2(x,y,z))$.

Оператор ограниченной минимизации. Пусть дан $(n+1)$-местный предикат $P(x_1,x_2,\ldots,x_n,y)$ на множестве N_0. Определим функцию $f(x_1,x_2,\ldots,x_n)$ следующим образом:

$$f(x_1,x_2,\ldots,x_n)=\begin{cases} y, \text{ если } y\le k,\ P(x_1,x_2,\ldots,x_n,y) \text{ истинно} \\ \quad \text{и для } \forall i < y\ P(x_1,x_2,\ldots,x_n,i) \text{ ложно} \\ k+1, \text{ если не существует } y \le k, \text{ такого что} \\ P(x_1,x_2,\ldots,x_n,y) \text{ истинно.} \end{cases}$$

Причем k в этом выражении может быть как константой, так и одной из переменных функции f. Другими словами, значение функции $f(x_1,x_2,\ldots,x_n)$ равно минимальному числу $y \le k$ (ограничена k), для которого предикат $P(x_1,x_2,\ldots,x_n,y)$ истинен или числу k+1, если такого y не существует. Говорят, что функция $f(x_1,x_2,\ldots,x_n)$ получается из предиката $P(x_1,x_2,\ldots,x_n,y)$ при помощи **оператора ограниченной минимизации**, при этом используется обозначение: $f(x_1,x_2,\ldots,x_n) = \mu_{y\le k} y\ P(x_1,x_2,\ldots,x_n,y)$.

Например, для одноместного предиката $P(y)=(y^2+y>6)$ значение функции f при ограничении единственной переменной $y \le 5$ будет равна $\mu_{y\le 5} y(y^2+y>6)=3$, так как, начиная с тройки, предикат $P(y)=(y^2+y > 6)$ будет истинным. Если переменная y будет ограничена единицей, то $\mu_{y\le 1} y(y^2+y>6)=2$, так как не существует $y \le 1$, при котором предикат P(y) будет истинным.

4.4. Для каждого из следующих предикатов найдите функцию, получаемую из них при помощи оператора ограниченной минимизации:

a) $P(x,y)=(x < 2(y+1))$; b) $P(x,y)=(x < y-2)$; c) $P(x,y)=(x < (y+1)^2)$;

d) $P(x,y)=(x < 2^{y+1})$.

Решение. а) Навесим на переменную y предиката P(x,y)=(x < 2(y+1)) оператор ограниченной минимизации. Получим одноместную функцию f(x)= $\mu y_{y\le x}$(x < 2(y+1)). Будем искать значения данной функции на множестве N_0 значений аргумента. Например, f(0) = $\mu y_{y\le 0}$(0 < 2(y+1))=0. И так далее, некоторые значения функции занесем в таблицу:

x	0	1	2	3	4	5	6
$f(x)$	0	0	1	1	2	2	3

Легко видеть, что полученные значения истинны для функции, которая находит целую часть от деления x на 2. Таким образом, f(x)=[x/2].

Утверждение. Если предикат $P(x_1,x_2,\ldots,x_n,y)$ примитивно рекурсивен, то и функция $\mu y_{i\le y}$ $P(x_1,x_2,\ldots,x_n,y)$ примитивно рекурсивна.

Так, например, функции из задания 4, полученные при помощи оператора ограниченной минимизации из примитивно рекурсивных предикатов, примитивно рекурсивные.

4.5. Докажите, что следующие функции примитивно рекурсивны:

a) f(x)=[a^x]; b) f(x)= [$\log_a x$]; c) $\mathrm{div}(x,y)=\begin{cases}[x/y], & y>0\\ x, & y=0;\end{cases}$

d) $\mathrm{rest}=\begin{cases}x \bmod y, & y>0\\ x, & y=0.\end{cases}$

Решение. c) Для начала определим примитивно рекурсивный предикат, с помощью которого в результате применения оператора ограниченной минимизации, может быть получена функция div(x,y). Число x делится на число y, если существует число z, что x=z·y. Но так как мы ищем целую часть от такого деления, необходимо, чтобы истина для предиката достигалась не только на одном конкретном z, но и на последующих, согласно определению оператора ограниченной минимизации.

Имеем предикат P(x,y,z) = (x < z·y), примитивная рекурсивность которого показана выше.

Рассмотрим функцию, полученную с помощью оператора ограниченной минимизации: f(x,y) = $\mu y_{z\le x}$ (x < z·y). Вычислим значение этой функции при некоторых значениях аргументов x и y:

f(1,1) = $\mu y_{z\le 1}$ (1 < z·1) = 2;
f(1,2) = $\mu y_{z\le 1}$ (1 < z·2) = 1;
f(2,1) = $\mu y_{z\le 2}$ (2 < z·1) = 2;
f(3,2) = $\mu y_{z\le 3}$ (3 < z·2) = 2;
f(4,2) = $\mu y_{z\le 4}$ (4 < z·2) = 3;
f(10,3) = $\mu y_{z\le 10}$ (10 < z·3) = 4;
f(0,0) = $\mu y_{z\le 0}$ (0 < z·0) = 1;
f(1,0) = $\mu y_{z\le 1}$ (1 < z·0) = 2;
f(3,0) = $\mu y_{z\le 3}$ (1 < z·0) = 4.

Легко видеть, что значение функции f(x,y) на единицу больше, чем необходимой для доказательства примитивной рекурсивности функции div(x,y). Таким образом, функция $\mathrm{div}(x,y) = \mu y_{z\le x}(x < z\cdot y) \dot{-} 1$ будет примитивно рекурсивной в силу примитивной рекурсивности усеченной разности.

Оператор минимизации. Частично рекурсивные и общерекурсивные функции. Пусть дан (n+1)-местный предикат $P(x_1,x_2,\ldots,x_n,y)$ на множестве N_0. Определим функцию $f(x_1,x_2,\ldots,x_n)$ следующим образом

$$f(x_1,x_2,\ldots,x_n)=\begin{cases} y, \text{ если } P(x_1,x_2,\ldots,x_n,y) \text{ истинный} \\ \quad \text{и для } \forall i < y \ P(x_1,x_2,\ldots,x_n,i) \text{ ложный} \\ \text{Ø, если не существует } y\in N_0, \text{ такого что} \\ \quad P(x_1,x_2,\ldots,x_n,y) \text{ истинный.} \end{cases}$$

Говорят, что функция $f(x_1,x_2,\ldots,x_n)$ получается из предиката $P(x_1,x_2,\ldots,x_n,y)$ при помощи **оператора минимизации**, при этом используется обозначение:

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

Таким образом, оператор минимизации позволяет определить частичную, т.е. не всюду определенную функцию (случай Ø). Например, разность f(x) = x-1 определена только на натуральных числах. Действительно, функцию можно записать при помощи оператора минимизации: $x-1 = \mu y(y+1=x)$. В точке x=0 значение функции не существует, так как нет такого $y\in N_0$, что предикат (y+1 = 0) будет истинным.

Функция называется частично рекурсивной, если она может быть получена из исходных с помощью конечного числа применения операторов суперпозиции, примитивной рекурсии и оператора минимизации. Если функция всюду определена и частично рекурсивна, то она называется **общерекурсивной**.

Функция f(x)=x-1 частично рекурсивна, так как получена с помощью оператора минимизации, примененного к примитивно рекурсивному предикату (y+1=x), или через исходные $(S(I_2^2) = I_2^1)$.

Существуют нигде не определенные частично рекурсивные функции. Например, $f(x)=\mu y(x+y+1=0)$.

4.6. Докажите, что следующие функции не всюду определенные частично рекурсивные:

a) f(x)=x/2; b) $f(x)=\sqrt{x}$; c) f(x,y)=x-y; d) f(x,y)=x/y.

Решение. c) Функция f(x,y)=x-y определена только при $x \ge y$. Разность x-y существует, если имеется такое число z, что x=z+y. Таким образом данную функцию можно получить с помощью оператора минимизации из примитивно рекурсивного предиката: $x-y=\mu z(x=z+y)$. Действительно, если x=1 и y=2, то не существует такого $z\in N_0$, при котором предикат (1=z+2) будет истинным. Ответ можно оформить с помощью исходных: $x-y=\mu z(I_3^1 = I_3^2 + I_3^2)$.

Очевидно, класс частично рекурсивных функций шире класса примитивно рекурсивных функций. Частично рекурсивные функции есть все исчерпывающиеся функции, поддающиеся вычислению с помощью некоторого алгоритма. Например, если известен алгоритм, описанный на одном из естественных языков или в виде программы для компьютера, то можно сказать, что эта функция частично рекурсивна.

Подобно тезису Тьюринга и принципу нормализации Маркова, в теории рекурсивных функций выдвигается аналогичная гипотеза, которую зовут **тезис Чёрча**: Всякая алгоритмически вычислимая функция является частично рекурсивной.

Утверждение. Функция частично рекурсивна тогда и только тогда, когда она вычислима по Тьюрингу или вычислима по Маркову. Другими словами, класс функций частично рекурсивных совпадает с классом функций, вычислимых по Тьюрингу, и с классом функций, вычислимых по Маркову.

Совпадение этих трех классов вычислимых функций, в основе которых лежат совершенно разные подходы к формализации понятия вычислимости функции, говорит о том, что эти подходы оказались состоятельными, и служат подтверждением того, что как принцип нормализации Маркова и тезис Тьюринга, так и тезис Чёрча имеют все права на признание.

5. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

5.1 Тренажёр Маркова

В настоящее время в сети Интернет можно найти программы-эмуляторы нормальных алгоритмов. Одна из них, учебная модель универсального исполнителя – тренажёр «Нормальные алгорифмы Маркова». Программа доступна на сайте К. Ю. Полякова «Преподавание, наука и жизнь» по адресу kpolyakov.narod.ru.

Вид программы представлен на рисунке 1. В верхней части окна находится поле редактора, в которое можно ввести условие задачи. Система подстановок, задающая НАМ, набирается в виде таблицы. Программа может выполняться непрерывно или по шагам. Существует возможность вызова протокола работы алгоритма.

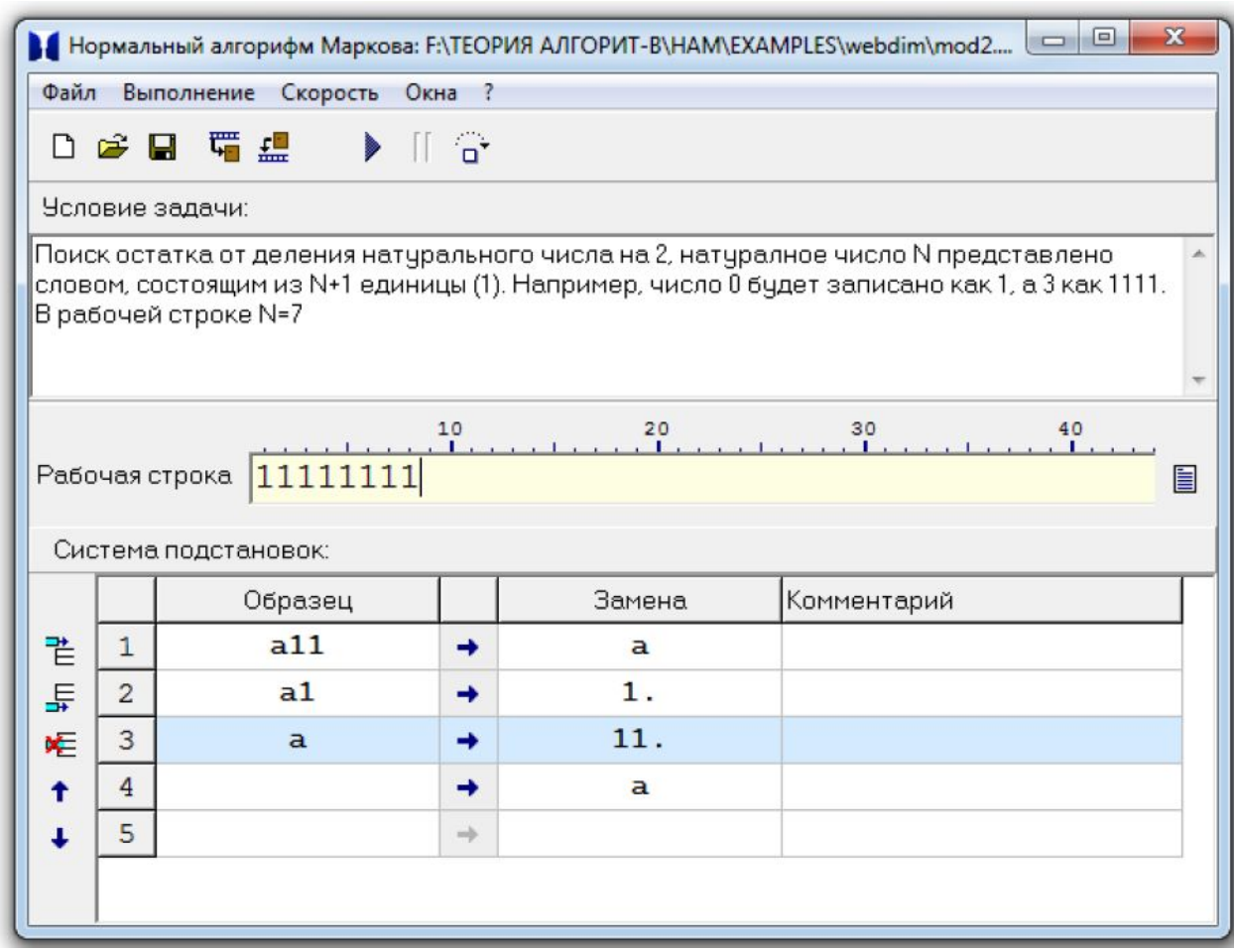


Рис. 1. Тренажёр «Нормальные алгорифмы Маркова»

5.2 Индивидуальные задания:

Пример1 (вставка и удаление символов)

A={a,b,c,d}. В слове **P** требуется заменить первое вхождение подслова **bb** на **ddd** и удалить все вхождения символа c.

Например: abbcabbca → adddabba

Пример2 (перестановка символов)

A={a,b}. Преобразовать слово **P** так, чтобы в его начале оказались все символы **a**, а в конце – все символы **b**.

Например: babba → aabbb

Пример3 (использование спецзнака)

A={a,b}. Удалить из непустого слова **P** его первый символ. Пустое слово не менять.

Пример4 (фиксация спецзнаком заменяемого символа)

A={0,1,2,3}. Пусть **P** – непустое слово. Трактуя его как запись неотрицательного целого числа в четверичной системе счисления, требуется получить запись этого же числа, но в двоичной системе.

Например: 0123 → 00011011

Пример5 (перемещение спецзнака)

A={a,b}. Требуется приписать символ **a** к концу слова **P**.

Например: bbab → bbaba

Пример6 (смена спецзнака)
A={a,b}. В слове **P** заменить на **aa** последнее вхождение символа **a**, если такое есть. Например: bababb → babaabb

Пример7 (перенос символа через слово)
A={a,b}. Перенести в конец непустого слова **P** его первый символ. Пустое слово не менять.
Например: bbaba → babab

Пример8 (использование нескольких спецзнаков)
A={a,b}. Удвоить слово **P**, т.е. приписать к **P** (слева или справа) его копию.
Например: abb → abbabb

5.3 Тренажёр Тьюринга

На сайте К.Ю. Полякова «Преподавание, наука и жизнь» имеется бесплатная программа-тренажёр «Машина Тьюринга», которая позволяет отработать навык составления программ на машине Тьюринга.

На рисунке 2 представлена программа, правильно вычисляющая числовую функцию f(x)= x mod 2, которая находит остаток от деления натурального x на 2 (см. зад. 14d).

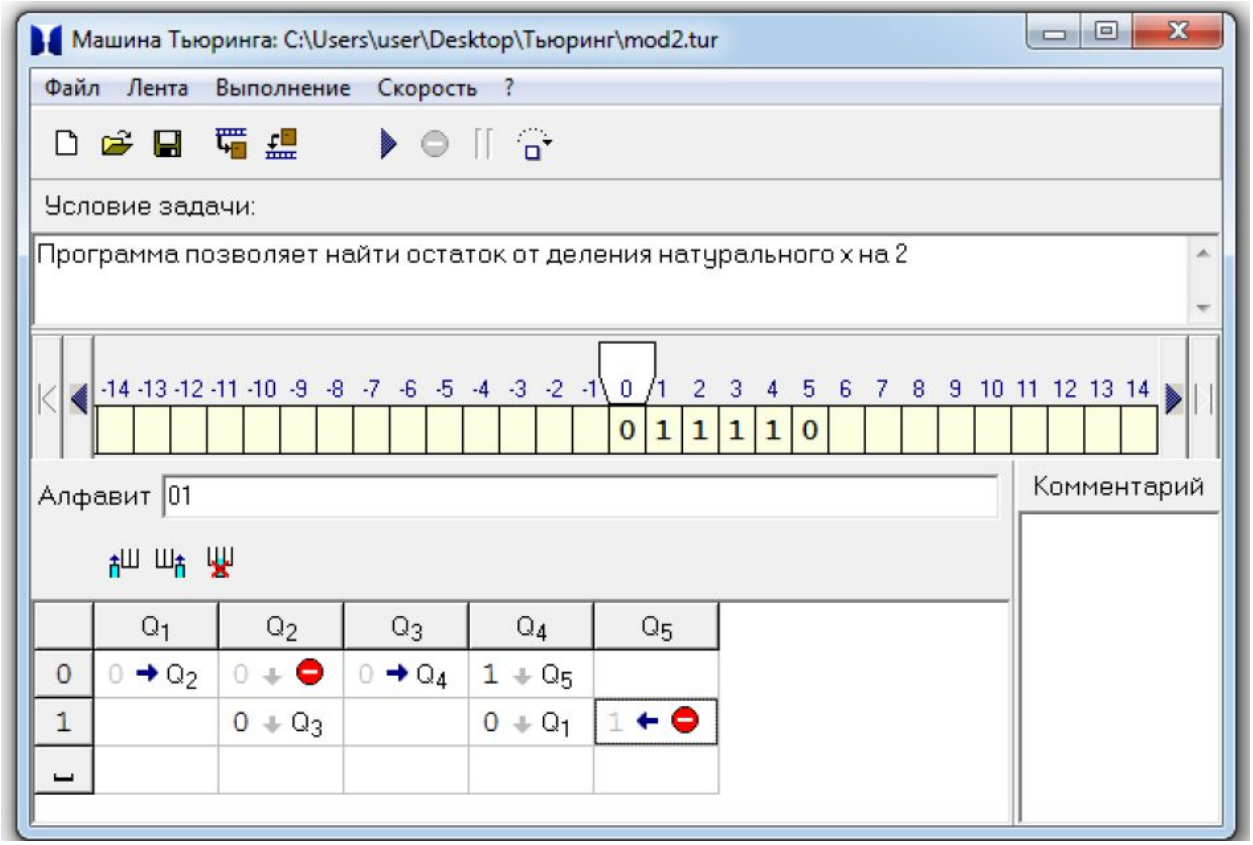


Рис. 2. Тренажёр «Машина Тьюринга»

Команды программы МТ набираются в виде таблицы. В первом столбце записаны символы алфавита, он заполняется автоматически. В первой строке перечисляются все возможные состояния. Добавить и удалить столбцы таблицы (состояния) можно с помощью кнопок, расположенных над таблицей. Программа может выполняться непрерывно или по шагам. Существует возможность сохранять написанные программы в файлах.

5.4 Индивидуальные задания

Пример1 (перемещение автомата, замена символов)

A={0,1,2,3,4,5,6,7,8,9}. Пусть **P** – непустое слово; значит, **P** – это последовательность из десятичных цифр. Требуется получить на ленте запись числа, которе на 1 больше числа **P.**

Пример2 (анализ символов)

A={a,b,c}. Перенести первый символ непустого слова **P** в его конец.

Пример3 (сравнение символов, стирание слова)

A={a,b,c}. Если первый и последний символы (непустого) слова **P** одинаковы, тогда это слово не менять, а иначе заменить его на пустое слово.

Пример4 (удаление символа из слова)

A={a,b}. Удалить из слова **P** его второй символ, если такой есть.

Пример5 (сжатие слова)

A={a,b,c}. Удалить из слова **P** первое вхождение символа **a**, если такое есть.

Пример6 (вставка символа в слово)

A={a,b,c}. Если **P** – непустое слово, то за его первым символом вставить символ **a**.

Пример7 (раздвижка слова)

A={a,b,c}. Вставить в слово **P** символ **a** за первым вхождением символа **c**, если такое есть.

Пример8 (формирование слова на новом месте)

A={a,b,c}. Удалить из **P** все вхождения символа **a.**

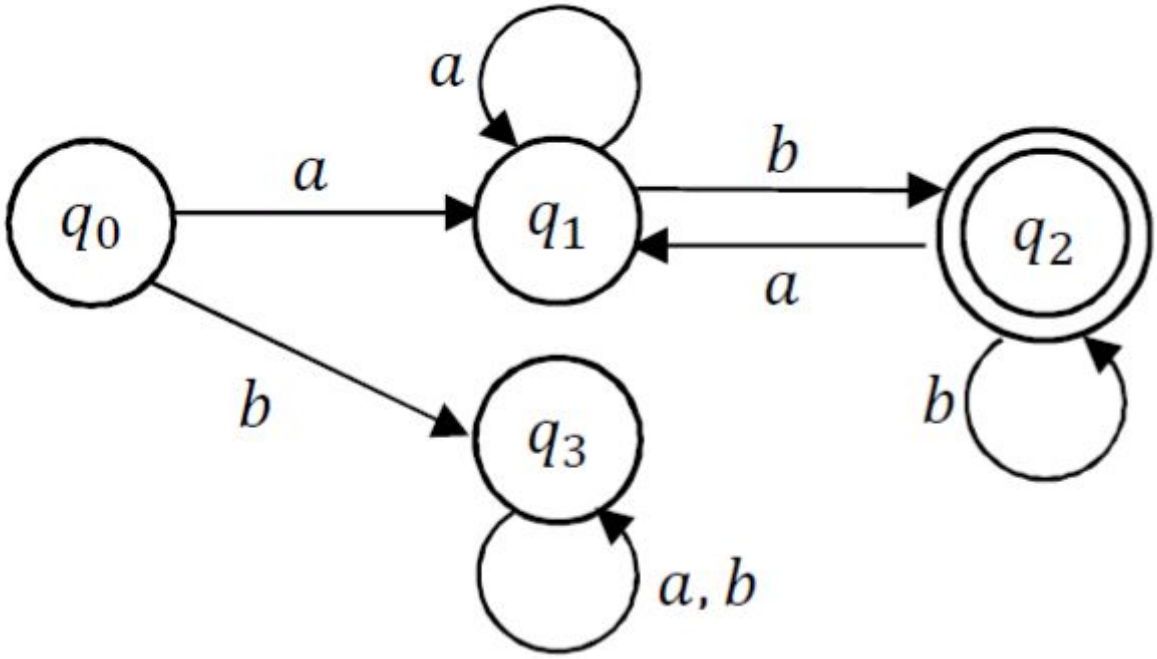
Пример9 (фиксирование места на ленте)

A={a,b}. Удвоить слово **P**, поставив меду ним и его копией знак =.

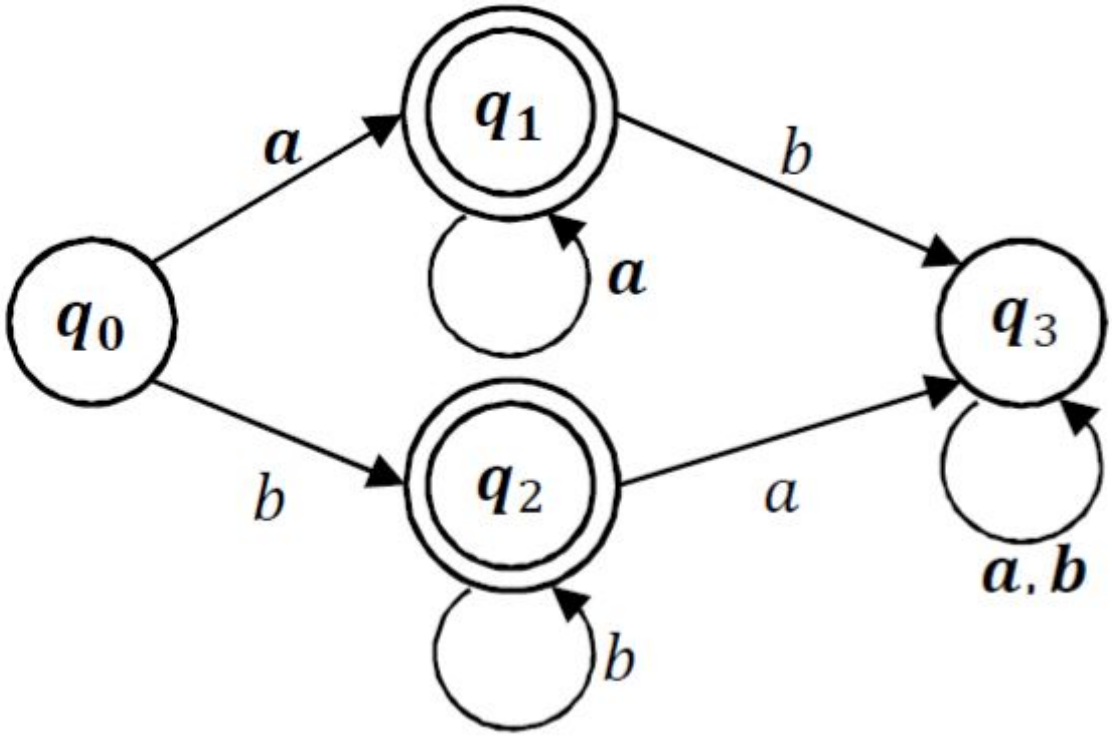
ОТВЕТЫ

1.1. Диаграмма автомата:

b)



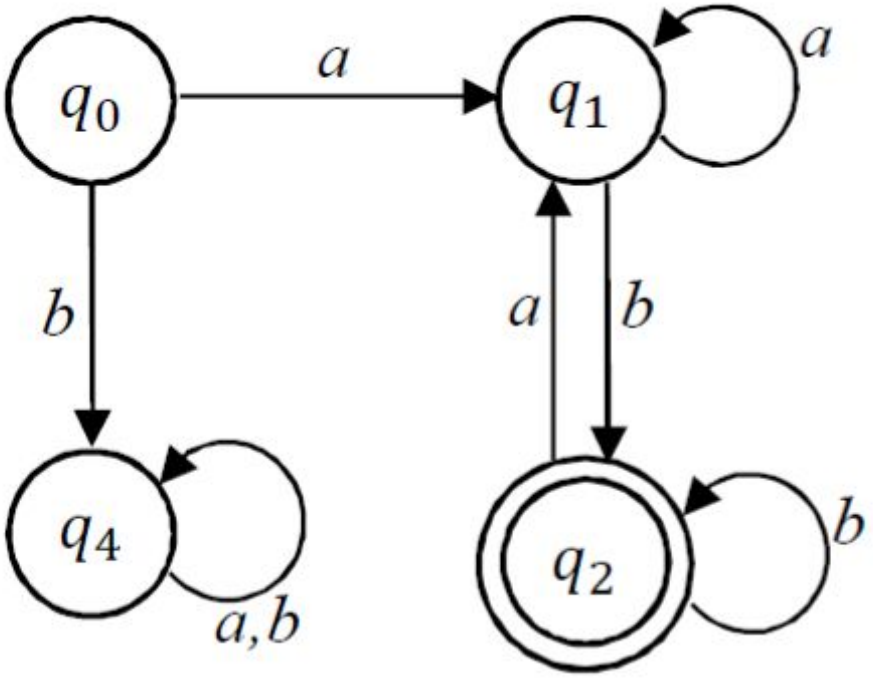
c)



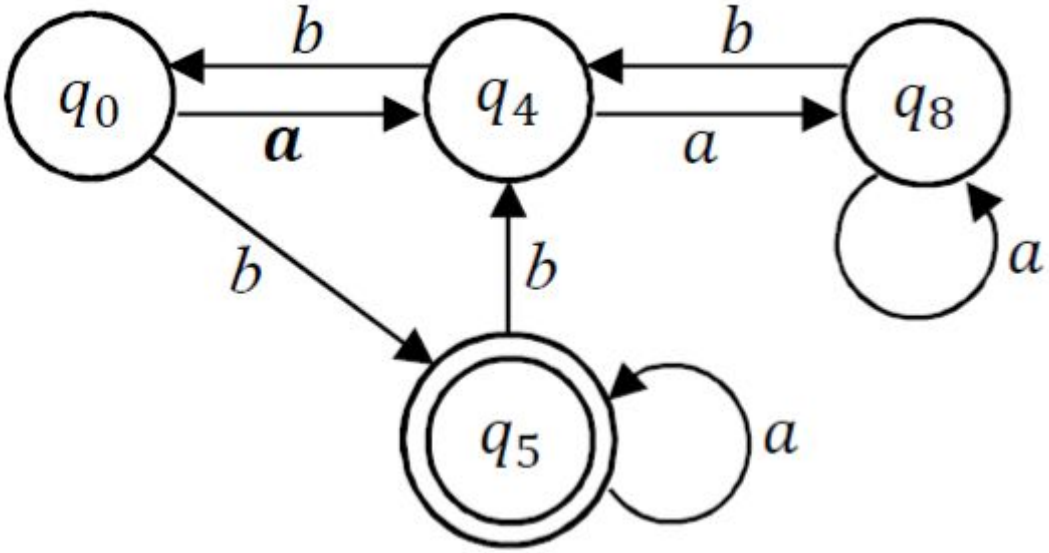
1.2. С помощью диаграммы изображен автомат, который в алфавите ∑={a,b} распознает: a) только слова a и ba; b) слова, состоящие из последовательностей символов ab (ab, abab,…); c) слова, начинающиеся и оканчивающиеся символом a.

1.3. Диаграмма автомата:

b)



c)



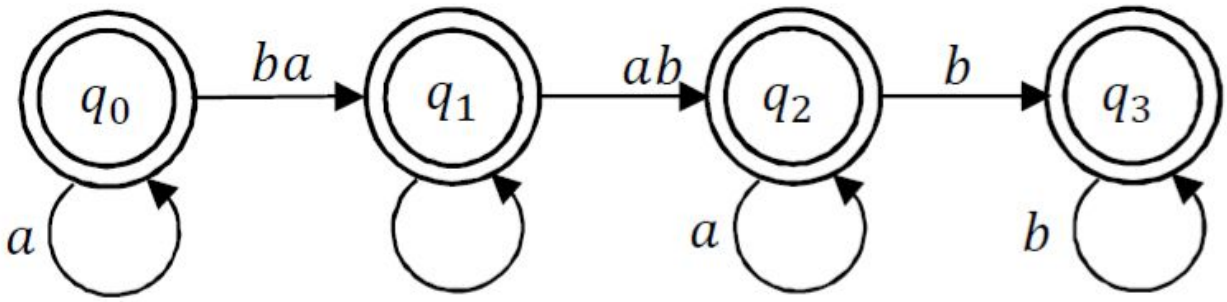
1.4. a) L(A)={ab, abab, ababab, ab…ab,…};
b) L(A)= {b, bb, bb…b,…, aa, aab, aabb, aabb…b,…};
d) L(A)={αaaa, αaab, αaba, αbaa, αbab| $\alpha \in \sum^*$}.

1.5. a) $L_1 \cup L_2$={a,b}; $L_1 \cdot L_2$={ab}; $L_2 \cdot L_1$={ba}; L_1^*={e,a,aa,aaa,…}=a^*;
L_2^*={e,b,bb,bbb,…}=b^*.
c) $L_1 \cup L_2$={a,b,ab,ba}; $L_1 \cdot L_2$={aba,abb,baa,bab}; $L_2 \cdot L_1$={aab,aba,bab,bba};
L_1^*=$(ab \cup ba)^*$; L=$(ab \cup ba)^*$; L_2^*=$(a \cup b)^*$.
d) $L_1 \cup L_2$={a,b,ab,ba,aab,baa};
$L_1 \cdot L_2$={aaba,aabb,aabab,aabba,baaa,baab,baaab,baaba};
$L_2 \cdot L_1$={aaab,abaa,baab,bbaa,abaab,abbaa,baaab,babaa};

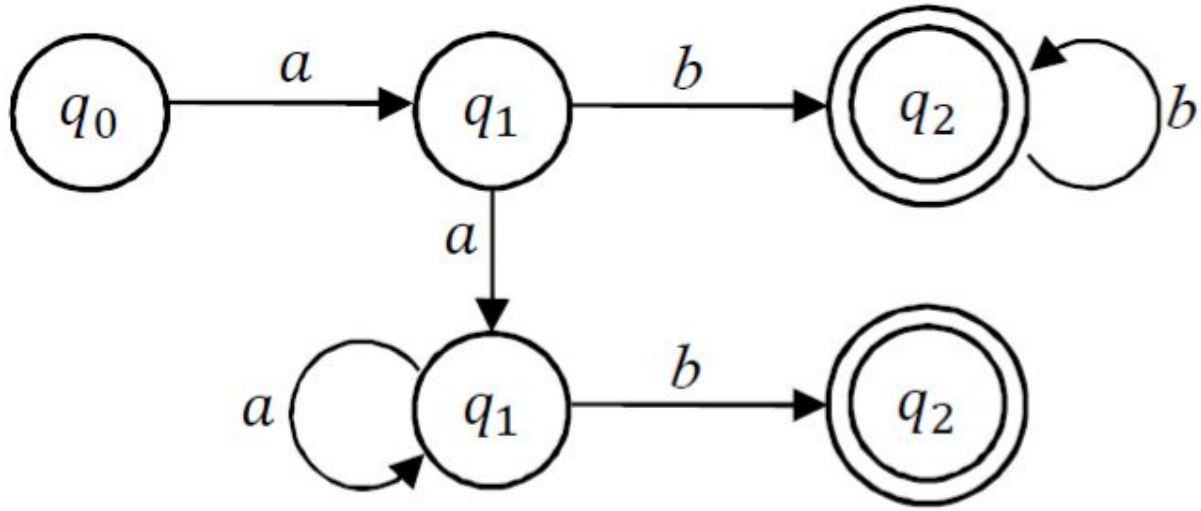
$L_1^*=(aab \cup baa)^*$; $L_2^*=(a \cup b \cup ab \cup ba)^*$.

1.6. Диаграмма автомата:

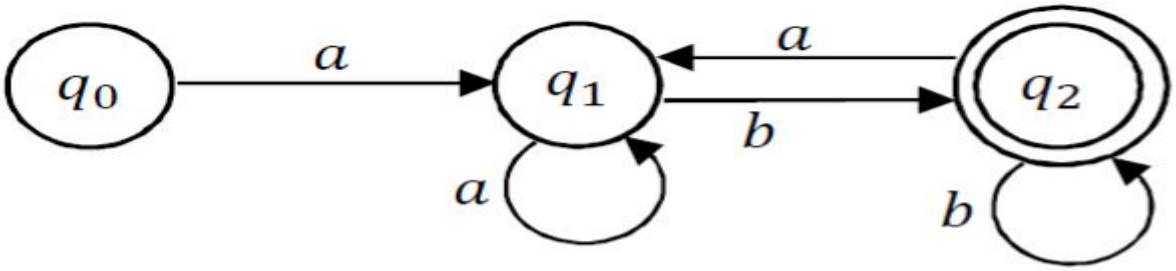
b)



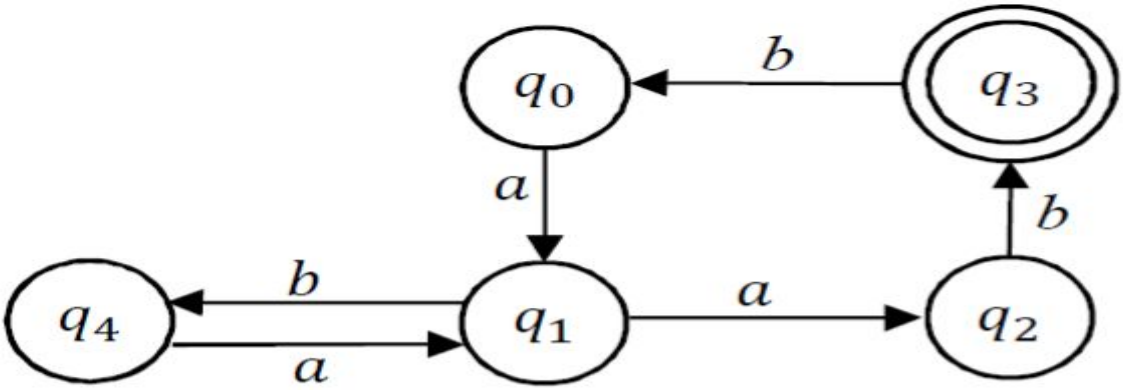
c)



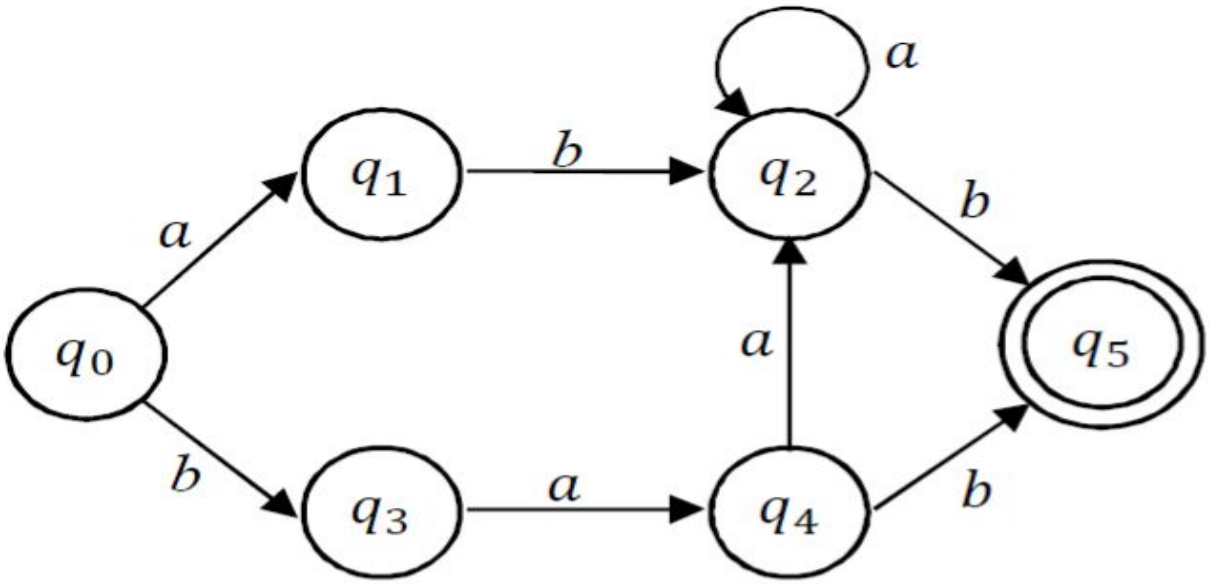
e)



f)



g)



2.1. d) Результат однократного применения: abaabbb. Дальнейшее применение: ababb.

e) Подстановка применима один раз, результат: abaab.

f) bbbabbb; g) bababbbb, bbabbabbb; h) aababaabbb, aabaababab, aaababaababb, aaabaababaabb,… При многократном применении подстановки происходит зацикливание. i) abaabaa.

2.2. c) Результат однократного применения: 1011111100. Дальнейшее применение: 1011111111. d) 10111001100, 101111001100, 101111001100…, (зацикливание); e) 11101100, 111110; f) 10011100, 100101.

2.4. b) Вычисление слова baba представим следующей последовательностью: γ_0=baba, γ_1=cbaba, γ_2=bcaba, γ_3=baacba, γ_4=baabca, γ_5=baabaac, γ_6=baabaa. c) aabbaab; d) bbb; e) baabaaaab; f) aaaaaabaa. Данный алгоритм преобразует каждое слово в алфавите ∑={a,b} в слово с удвоенными символами a.

2.5. Алгоритм не применим только к словам b) и f).

2.6. Схема нормального алгоритма примет вид:

b)
$$\begin{cases} a \to \\ b \to \\ \to .ab \end{cases}$$

d)
$$\begin{cases} ca \to aac \\ cb \to bbc \\ c \to . \\ \to c \end{cases}$$

e)
$$\begin{cases} ca \to bc \\ cb \to ac \\ c \to . \\ \to c \end{cases}$$

f)
$$\begin{cases} ca \to adac \\ cb \to bdbc \\ daa \to ada \\ dab \to bda \\ dba \to adb \\ dbb \to bdb \\ d \to \\ c \to . \\ \to c \end{cases}$$

2.7. Схема нормального алгоритма примет вид:

a) $\{ 1 \to .111$

d) $\{ 11 \to .1$

e)
$$\begin{cases} a11 \to 1a \\ a1 \to .1 \\ a \to . \\ \to a. \end{cases}$$

f)
$$\begin{cases} a11 \to a \\ a1 \to .1 \\ a \to .11 \\ \to a \end{cases}$$

g)
$$\begin{cases} a111 \to a11 \\ a11 \to .11 \\ 1 \to .1 \\ \to a \end{cases}$$

i)
$$\begin{cases} 1-1 \to - \\ -\to .1 \end{cases}$$

j)
$$\begin{cases} 1-1 \to - \\ -1 \to -1 \\ -\to .1 \end{cases}$$

k)
$$\begin{cases} 1*1 \to *a \\ a1 \to 1a \\ 1* \to * \\ *1 \to * \\ * \to \\ a \to 1 \end{cases}$$

m)
$$\begin{cases} 1*11 \to b*1 \\ *1 \to b* \\ bb* \to \\ 1b \to ba1 \\ ab \to ba \\ b \to \\ a \to 1 \\ \to .1 \end{cases}$$

n)
$$\begin{cases} 1-1 \to - \\ -\to 1 \\ a11 \to a \\ a \to .1 \\ a \to .11 \\ \to a \end{cases}$$

3.1. b) bbbaa; c) babaab; d) aabbbabb; e) abbabaabaa. Данная машина Тьюринга переставляет первый символ любого слова в алфавите {a,b} в конец слова. Для того чтобы заключительная конфигурация имела стандартное положение, необходимо осуществить сдвиг головки влево до первого символа слова. Для этого исправим две последние команды и допишем новые. Всю программу кратко можно представить так: $q_1aa_0q_2$, $q_1ba_0q_3$, $q_2a_0Rq_4$, $q_3a_0Rq_5$, q_4aRq_4, q_4bRq_4, q_5aRq_5, q_5bRq_5, $q_4a_0aq_6$, $q_5a_0bq_6$, q_6aLq_6, q_6bLq_6, $q_6a_0Rq_0$.

3.2. Программа в виде таблицы:

Σ \ Q	q_1	q_2
a_0	a_0q_0	Rq_1
a	a_0q_2	

a) a_0 (все ячейки ленты пустые), переходы между конфигурациями: $q_1aaa \to q_2a_0aa \to q_1aa \to q_2a_0a \to q_1a \to q_2a_0 \to q_1a_0 \to q_0a_0$; b) a, переходы между конфигурациями: $q_1aaa_0a \to q_2a_0aa_0a \to q_1aa_0a \to q_2a_0a_0a \to q_1a_0a \to q_0a_0a$; c) aaaa, переходы между конфигурациями: $q_1aa_0aaaa \to q_2a_0a_0aaaa \to q_1a_0aaaa \to q_0a_0aaaa$; d) a.

3.3. a) Программа в виде последовательности команд: q_1abq_2; q_1baq_2; q_2aRq_2; q_2bRq_2; $q_1a_0Lq_3$; q_3aLq_3; q_3bLq_3; $q_3a_0Rq_0$; b) baab; aabb; bbabba; bbbb; aaabbb; babababa. c) Данная машина Тьюринга выполняет замену всех символов a слова в алфавите {a,b} на символ b и наоборот.

3.4. Программа в виде таблицы:

Σ \ Q	q_1	q_2	q_3	q_4
0		$1q_3$		$1q_0$
1	Lq_2		Lq_4	

a) 1111, последовательность конфигураций, возникающих на ленте на каждом шаге работы, изобразим графически:

1) q_1: | | | | 1 | 1 | |

2) q_2: | | | 0 | 1 | 1 | |

3) q_3: | | | 1 | 1 | 1 | |

4) q_4: | | 0 | 1 | 1 | 1 | |

5) q_0: | | 1 | 1 | 1 | 1 | |

b) 11111; c) 111111; d) $\underbrace{11\ldots1}_{n+2}$.

Заключительная конфигурация находится в стандартном положении. Машина Тьюринга к единичкам на ленте добавляет еще две. Для данной МТ можно сконструировать ей равносильную с меньшим числом состояний:

Σ \ Q	q_1	q_2
0	$1q_2$	$1q_0$
1	Lq_1	Lq_2

3.5. Программа в виде последовательности команд: q_110q_1, q_10Rq_2, q_110q_2, q_20Rq_0.

a) 1; b) 0 (нет единиц на ленте); c) машина Тьюринга не останавливается; d) $\underbrace{11\ldots1}_{n-2}$.

Машина удаляет две единички на ленте при условии, что они там есть. Если на ленте менее двух единичек, то машина Тьюринга не останавливает свою работу, т.е. она не применима к словам 1 и 0.

3.6. b) 1111; c) 111; d) $\underbrace{11\ldots1}_{n+m}$.

Нетрудно видеть, что МТ реализует операцию сложения: в результате ее работы на ленте будет записано столько единиц, сколько их было записано по обе стороны от звездочки перед началом работы.

3.7. a) 111; b) 11; c) 0 (все ячейки ленты пустые); d) $\underbrace{11\ldots1}_{n}$.

Машина Тьюринга стирает все единицы, расположенные справа от звездочки перед началом работы.

3.10. Программа МТ в виде таблицы:

Σ \ Q	q_1	q_2	q_3	q_4
a_0	Lq_2		aq_0	bq_0
a	Rq_1	a_0q_3	Lq_3	Lq_4
b	Rq_1	a_0q_4	Lq_3	Lq_4

3.11. Программа МТ в виде набора команд:

$q_1aa_0q_2$	q_7aRq_8	$q_{11}aLq_{13}$	$q_{14}a_0Rq_7$	$q_{18}aLq_{18}$
$q_1ba_0q_3$	q_7bRq_8	$q_{11}bLq_{13}$	$q_8a_0Lq_{15}$	$q_{18}bLq_{18}$
$q_2a_0Lq_4$	q_8aLq_9	$q_{12}aLq_{12}$	$q_{15}aa_0q_{16}$	$q_{19}aLq_{19}$
$q_3a_0Lq_5$	q_8bLq_9	$q_{12}bLq_{12}$	$q_{15}ba_0q_{17}$	$q_{19}bLq_{19}$
$q_4a_0aq_6$	$q_9aa_0q_{10}$	$q_{13}aLq_{13}$	$q_{16}a_0Lq_{16}$	$q_{18}a_0aq_0$
$q_5a_0bq_6$	$q_9ba_0q_{11}$	$q_{13}bLq_{13}$	$q_{17}a_0Lq_{17}$	$q_{19}a_0bq_0$
q_6aRq_6	$q_{10}a_0Lq_{10}$	$q_{12}a_0aq_{14}$	$q_{16}aLq_{18}$	
q_6bRq_6	$q_{10}aLq_{12}$	$q_{13}a_0bq_{14}$	$q_{16}bLq_{18}$	
$q_6a_0Rq_7$	$q_{10}bLq_{12}$	$q_{14}aRq_{14}$	$q_{17}aLq_{19}$	
$q_7a_0Rq_7$	$q_{11}a_0Lq_{11}$	$q_{14}bRq_{14}$	$q_{17}bLq_{19}$	

3.12. Программа МТ: a) q_110q_2, q_20Rq_1, q_100q_0; b) q_11Lq_1, q_101q_0; c) q_110q_1, q_10Rq_0; d) q_110q_2, q_100q_0, q_20Rq_3, q_310q_4, q_301q_0, q_40Rq_1.

3.13. Программа МТ:

a)

Σ \ Q	q_1	q_2	q_3	q_4	q_5	q_6
0		Lq_4	Rq_2	Lq_4	Lq_6	Rq_0
1	Rq_1	$0q_3$				Lq_6
*	Rq_2			$0q_5$		

b)

Σ \ Q	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
0		Lq_3		Lq_5		Rq_7		Rq_1	Lq_{10}	Rq_0
1	Rq_1	Rq_2	$0q_4$		Lq_5	Lq_6	$0q_8$			Lq_{10}
*	Rq_2		$0q_9$		Lq_6					

c)

Σ \ Q	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
0				Lq_5					Lq_{10}	
1	aq_2		Rq_3	Rq_4	aq_6		Lq_7	Lq_8		
a		Rq_3		Lq_5		Lq_7		Rq_1	Rq_9	$0q_9$
*	Lq_{13}		Rq_4		Rq_9		Lq_8		$0q_{11}$	

Σ \ Q	q_{11}	q_{12}	q_{13}	q_{14}	q_{15}	q_{16}	q_{17}	q_{18}	q_{19}
0	Lq_{12}	Rq_0	Rq_{14}		Rq_{14}	Rq_{17}	Lq_{18}	Rq_0	
1		Lq_{12}					Rq_{17}	Lq_{18}	Lq_{18}
a		$1q_{12}$	Lq_{13}	$0q_{15}$			Rq_{17}	$1q_{19}$	
*				$0q_{16}$					

3.16. f) Программа МТ, правильно вычисляющая функцию $f(x) = sgn(x)$, такова: $q_1 0 R q_2$, $q_2 1 R q_3$, $q_2 0 L q_0$ (для $x = 0$), $q_3 1 0 q_4$, $q_4 0 R q_3$, $q_3 0 L q_5$, $q_5 0 L q_5$, $q_5 1 L q_0$.

i) Функция $f(x, y) = x - y$ определена на тех значениях аргументов, при которых $x \geq y$. Только в этом случае машина должна на выходе давать результат $x - y$, иначе она должна работать вечно. За основу возьмем программу зад. 13b:

Σ \ Q	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
0	Rq_2	Rq_3	Lq_4	Lq_{10}	Lq_6	Lq_7	Rq_8	Lq_8	Rq_2	$0q_0$
1		Rq_2	Rq_3	$0q_5$		Lq_6	Lq_7	$0q_9$		Lq_{10}

Указание. Машина стирает за цикл по одной единички справа и слева до тех пор, пока они там есть, осуществляя переход конфигураций $q_1 01^x 01^y 0 \to q_i 01^{x-1} 01^{y-1} 0$, $i > 1$. Если число единиц x больше или равно числа единиц y, то машина попадет в ситуацию $q_{10}0$ и работа будет завершена конфигурацией $q_0 01^{x-y} 0$. Если число единиц x меньше числа единиц y, то машина попадет в ситуацию $q_8 0$ и, согласно команде $q_8 0 L q_8$, будет работать вечно.

4.1. b) Применим функцию следования $S(x)$ k-раз: $f(x) = \underbrace{S(S(\dots S}_{k}(x)\dots)) = x + k$. d) $f(x) = \underbrace{x + \dots + x}_{k} = kx$. f) $f(x) = \underbrace{x + \dots + x}_{k} = x^k$.

4.2. a) $g = f(0) = 0! = 1$, $h(x,y) = f(x+1) = (x+1)! = x! \cdot (x+1) = f(x) \cdot (x+1) = (x+1)y$; c) $g(x) = f(x,0) = x = I_1^1$, $h(x,y,z) = (x \dot{-} y) \dot{-} 1 = I_1^1 \dot{-} 1$; d) $|x-y| = (x \dot{-} y) + (y \dot{-} x)$; e) $max(x,y) = y + (x \dot{-} y)$; f) $min(x,y) = x \dot{-} (x \dot{-} y)$; h) $\overline{sgn}(x) = 1 \dot{-} x$.

4.3. b) $\chi_P(x,y) = \overline{sgn}|x-y|$; c) $\chi_P(x,y) = \overline{sgn}(x \dot{-} y)$; f) $(x \cdot y = z) = Q\big(f_1(x,y,z), f_2(x,y,z)\big)$, где предикат $Q(x,y) = (x = y)$ и функции $f_1(x,y,z) = x \cdot y$, $f_2(x,y,z) = z$ являются примитивно рекурсивными.

4.4. b) $\mu y_{y \le x}(x < y - 2) = x + 3$; c) $\mu y_{y \le x}[x < (y+1)^2] = [\sqrt{x}]$; d) $\mu y_{y \le x}\left(x < 2^{y+1}\right) = [\log_2 x]$.

4.5. a) $[a^x] = \mu y_{y \le x}[x < \log_a(y+1)]$; c) $[\log_a x] = \mu y_{y \le x}(x < a^{y+1})$; d) $rest(x,y) = x \dot{-} y \cdot div(x,y)$.

4.6. a) $x/2 = \mu y(x = 2y)$; b) $\sqrt{x} = \mu y(x = y^2)$; d) $x/y = \mu z(x = z \cdot y)$.

Список литературы

1. Алферова З.В. Теория алгоритмов. – М.: Статистика, 1973. – 164с.
2. Мальцев А.И. Алгоритмы и рекурсивные функции. – М.: Наука, 1960. – 392с.
3. Марков А.А., Нагорный Н.М. Теория алгорифмов. – М.: ФАЗИСТ, 1996. – 448с.