Косовская Татьяна Матвеевна

МАШИНЫ ТЬЮРИНГА

МАТЕМАТИЧЕСКОЕ ПОНЯТИЕ АЛГОРИТМА

Понятие алгоритма прочно вошло в жизнь математиков, а особенно людей, тесно связанных с вычислительной техникой. Зачастую мы даже не задумываемся, что же это такое, а используем как некое интуитивное понятие. Однако история возникновения этого понятия восходит к глубокой древности.

Термин алгоритм или алгорифм (записываемый латинскими буквами как *algorithm*) имеет в своем составе видоизмененное географическое название *Хорезм* и обязан своим происхождением великому средневековому ученому (приблизительно 783–850 гг.) Мухаммаду ибн Муссе аль Хорезми (то есть из Хорезма).

Первоначально под алгоритмом понимали произвольную строго определенную последовательность действий, приводящую к решению той или иной конкретной задачи. Еще с античных времен известны алгоритм Евклида нахождения наибольшего общего делителя двух натуральных чисел, алгоритм деления отрезка в заданном отношении с помощью циркуля и линейки, алгоритмы умножения и деления чисел (что было очень трудно до повсеместного использования позиционной системы счисления) и т. п.

Первоначально под алгоритмом понимали произвольную строго определенную последовательность действий, приводящую к решению той или иной конкретной задачи

В начале XX века были сформулированы задачи нахождения единого процесса решения ряда родственных задач с параметрами. Если с нахождением такого процесса все было более или менее ясно – достаточно предъявить такой процесс, который решает требуемую задачу, то что же делать, если процесс найти не удалось? Мы плохо искали или его не существует? Ответы на эти вопросы были особенно важны в связи с программой Д. Гильберта формализации всей математики.

Первые попытки дать математическое определение алгоритма привели приблизительно к следующим требованиям:

– *Определенность данных* (вид исходных данных строго определен).

– *Дискретность* (процесс разбивается на отдельные шаги).

– *Детерминированность* (результат каждого шага строго определен в зависимости от данных, к которым он применен).

– *Элементарность шагов* (переход на один шаг прост).

– *Направленность* (что считать результатом работы алгоритма, если следующий шаг невозможен).

– *Массовость* (множество возможных исходных данных потенциально бесконечно).

Несмотря на недостатки этого определения (например, что такое «*переход на один шаг прост*»?), это интуитивное определение может служить для решения многих задач. Однако в современной математике и информатике активно используются алгоритмы, не удовлетворяющие такому интуитивному определению, например недетерминированные вычисления, алгоритмы с оракулом, «зацикливающиеся» алгоритмы и т.п.

Для математического уточнения определения алгоритма, начиная с 30-х годов

XX века, были введены различные математические понятия алгоритма. Одним из наиболее важных для вычислительной техники оказалось понятие машины Тьюринга (или машины Тьюринга-Поста), введенное и опубликованное практически независимо Тьюрингом и Постом.

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Машина Тьюринга является математическим, а не техническим понятием. Однако те, кому ближе «железные» объекты, могут представлять ее как вычислительное устройство, имеющее потенциально бесконечную ленту, разделенную на ячейки (то есть в любой момент работы слева или справа к конечной ленте можно добавить недостающую ячейку, содержащую специальный символ, называемый пустым). На этой ленте могут быть записаны слова в некотором заранее фиксированном алфавите. По ленте может перемещаться пишущая–читающая головка, обозревающая одну из ячеек. В зависимости от состояния машины и содержимого обозреваемой ячейки, машина может, в соответствии с командой программы, изменить (или не изменять) состояние, заменить (или не заменять) содержимое обозреваемой ячейки, сдвинуть головку на одну ячейку влево, или вправо, или не сдвигать ее.

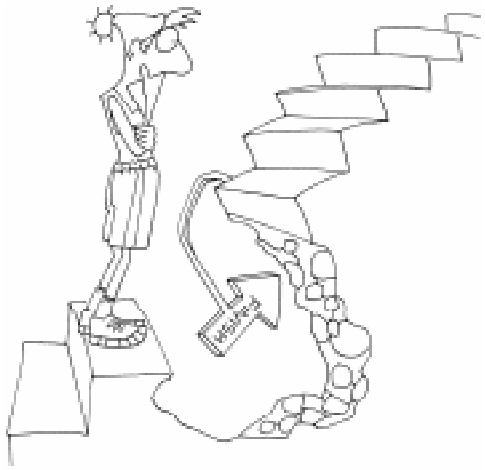
С математической точки зрения, машина Тьюринга задается двумя алфавитами и программой. Алфавит $A = \{a_1, \ldots, a_n\}$ – внешний алфавит символов (содержащий, в частности, пустой символ, который здесь мы будем обозначать посредством *). Алфавит $Q = \{q_0, q_1, \ldots, q_k\}$ – внутренний алфавит состояний. При этом q_1 называется начальным состоянием машины Тьюринга, а q_0 – ее заключительным состоянием.

Команда машины Тьюринга имеет вид

$$q_r\, a_i \to q_t\, S\, a_j,$$

где $S \in \{L, R, \_\}$ и обозначает, соответственно, сдвиг влево, вправо или отсутствие сдвига головки.

Эта команда читается следующим образом: «Если машина Тьюринга находится в состоянии q_r и в обозреваемой ячейке записан символ a_i, то этот символ заменяется



«переход на один шаг прост»

на a_j, головка производит сдвиг S, и машина Тьюринга переходит в состояние q_t».

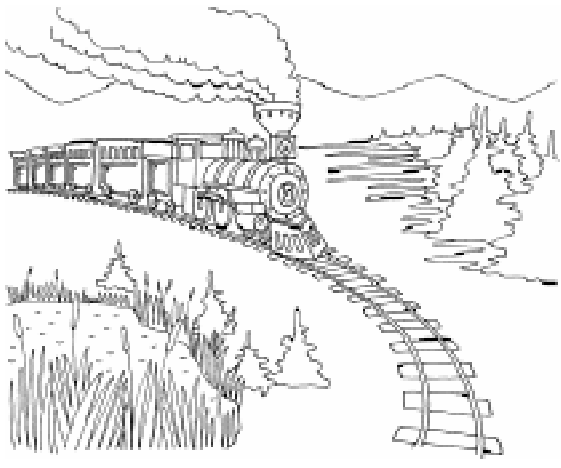
Две команды называются *согласованными*, если они либо имеют различные левые части, либо полностью совпадают. Это требование обеспечивает детерминированность вычисления.

Программой машины Тьюринга называется конечное непустое множество попарно согласованных команд.

Конфигурацией машины Тьюринга называется слово вида

$$b_1 \ldots b_{p-1}\, q_j\, b_p \ldots b_1,$$

где $b_1 \ldots b_{p-1}\, b_p \ldots b_1$ – слово в алфавите A,



...те, кому ближе «железные» объекты, могут представлять ее как вычислительное устройство, имеющее потенциально бесконечную ленту, разделенную на ячейки...

записанное на ленте (причем слева и справа от этого слова находятся только пустые ячейки ленты, на левом и правом концах этого слова может находиться не более одного пустого символа), машина находится в состоянии q_j и обозревает p-ый символ этого слова.

Для любителей технических устройств приведенная конфигурация соответствует следующей ситуации

*****	b_1	...	b_{p-1}	b_p	b_{p+1}	...	b_l	*****

$\uparrow$
q_j

Машина Тьюринга всегда начинает работу в начальной конфигурации вида q_1X, где X – исходные данные, а заканчивает в случае, когда пришла в заключительное состояние q_0.

Протоколом работы машины Тьюринга называется последовательность конфигураций, первая из которых является начальной, а каждая следующая получена из предыдущей в соответствии с одной из команд.

Машина Тьюринга *заканчивает* работу над данными X, если она пришла в состояние q_0. Результатом работы машины Тьюринга будет слово, записанное на ленте. При этом говорят, что машина Тьюринга применима к данным X. Если в процессе работы с данными X машина Тьюринга никогда не приходит в состояние q_0 или в процессе ее работы ни одна из команд программы не может быть применена, то говорят, что машина Тьюринга не применима к данным X.

КАК ЖЕ РАБОТАЕТ МАШИНА ТЬЮРИНГА?

Рассмотрим несколько примеров машин Тьюринга.

Пример 1.

Написать машину Тьюринга, вычисляющую сумму двух чисел, записанных в унарной системе счисления.

Напомним, что вид исходных данных для любого алгоритма должен быть строго определен. Поэтому нельзя написать машину Тьюринга, вычисляющую сумму двух чисел, пока не задана система счисления. Унарная система счисления – это так называемая единичная (или палочковая) система: каково число – столько единичек (палочек).

Начальная конфигурация машины Тьюринга будет $q_1$1...1+1...1 , где количество единиц в первом слагаемом равно x, а количество единиц во втором слагаемом равно y. Требуется, чтобы заключительной конфигурацией была $q_0$1... 1, где количество единиц равно $x + y$. Для этого разработаем план работы машины Тьюринга.

1. Сотрем первую единицу.

2. Будем сдвигать головку вправо, пока не увидим знак +, который заменим на 1.

3. Будем сдвигать головку влево, пока не увидим знак * (пустой символ).

4. Сдвинем головку на один символ вправо и остановимся.

Реализация каждого пункта этого плана требует свое собственное состояние, поэтому, записав команды, реализующие пункт плана, обязательно будем менять состояние машины Тьюринга.

1.1) $q_1\ 1 \rightarrow q_2\ *$
2.1) $q_2\ 1 \rightarrow q_2\ R\ 1$
2.2) $q_2\ + \rightarrow q_3\ 1$
3.1) $q_3\ 1 \rightarrow q_3\ L\ 1$
4.1) $q_3\ * \rightarrow q_0\ R\ *$

Запишем протокол работы этой машины Тьюринга, вычисляющей 2 + 3.

$q_1$11+111 (по 1.1) $\Rightarrow$
q_2 1+111 (по 2.1) $\Rightarrow$
1 q_2 +111 (по 2.2) $\Rightarrow$
1 q_3 1111 (по 3.1) $\Rightarrow$
q_3 11111 (по 3.1) $\Rightarrow$
q_3 *11111 (по 4.1) $\Rightarrow$
q_0 11111

В этой машине Тьюринга был использован алфавит {*,1,+}. Однако можно было бы использовать и алфавит {*,1}, при этом исходные данные разделяются знаком * , а в команде 2.2) вместо + следует поставить *.

Задание 1.

Написать машину Тьюринга, заменяющую в записи десятичной дроби знак «,» (запятая) на знак «.» (точка).

Уровень 1. Определить внешний алфавит машины Тьюринга, исходную и заключительную конфигурации. Составить план работы машины Тьюринга.

Уровень 2. Написать программу машины Тьюринга и протокол ее работы над числом x (при $x = 2{,}1$ и $x = 123, 728$).

Уровень 3. Проведите исследования, сколько конфигураций в процессе работы этой машины Тьюринга появится при работе над числом, имеющим вид

$$a_1 a_2 \ldots a_n , \; b_1 b_2 \ldots b_m$$

где $a_1 a_2 \ldots a_n b_1 b_2 \ldots b_m$ – десятичные цифры.

Пример 2.

Написать машину Тьюринга, вычисляющую сумму двух чисел, записанных в двоичной системе счисления.

Начальная конфигурация машины Тьюринга будет q_1 $X*Y$, где X и Y – двоичные записи натуральных чисел. Требуется, чтобы заключительной конфигурацией была q_0 Z, где Z – двоичная запись суммы. Для этого разработаем план работы машины Тьюринга.

1. «Добежим вправо» до разделяющего аргументы пустого символа *.

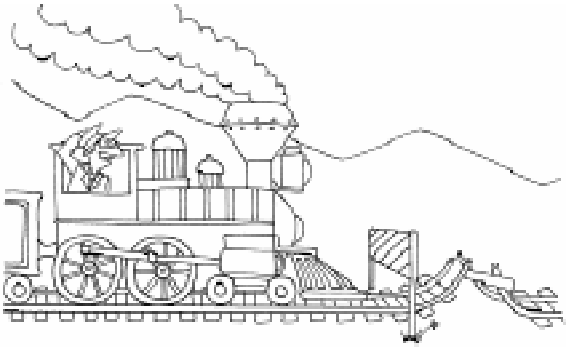
2. «Добежим вправо» до последней непомеченной цифры числа Y. (В начальный момент все цифры не помечены.)

3. Запомним эту цифру и пометим ее. Отметим, что машина Тьюринга может запоминать что-либо только номером состояния. Пометить цифру можно, например, заменив 0 на a, а 1 на b.

4. «Добежим влево» до последней непомеченной цифры числа X, запомним эту цифру и пометим ее.

5. «Добежим влево» до первой цифры числа X и отступим на одну клетку влево.

6. «Добежим влево» до первой цифры числа, полученного сложением просмотренных частей X и Y. Отступив на одну клетку влево, запишем сумму запомненных цифр, при этом, если складывались 1 + 1, то записываем 1 и запоминаем, что к сумме следующих двух цифр будет необходимо прибавить 1.



*«Добежим вправо» до разделяющего аргументы пустого символа *.*

7. «Добежим вправо» до *, стоящей после последней цифры числа вычисленной части суммы и перейдем к выполнению п. 1 нашего плана.

8. Если одно из слов (X либо Y) оказалось короче другого, то *, стоящую перед соответствующим словом будем запоминать для сложения как цифру 0.

9. Если оба аргумента полностью помечены, то сотрем помеченные цифры, переведем головку в начало результирующего слова и остановимся.

Как видно из этого плана, программа машины Тьюринга будет очень длинной и потребует большого количества состояний. Поэтому этот пример рассмотрим еще раз для другой модели машины Тьюринга.

Пример 3.

Написать программу машины Тьюринга, осуществляющую удвоение слова в алфавите $\{a, b\}$.

Начальная конфигурация машины Тьюринга имеет вид q_1X , где X – слово в алфавите $\{a, b\}$. Конечная конфигурация имеет вид q_0X*X.

План работы машины Тьюринга.

1. Запомним текущий символ слова и пометим его. Для запоминания символа a будем использовать состояние q_2, а для запоминания символа b – состояние q_3. Помечать символы будем, записывая вместо них соответственно α и β.

2. «Добежим вправо» до *, стоящей после исходного слова. Сдвинемся на одну ячейку вправо, изменив состояние q_2 на q_4, а состояние q_3 на q_5.

3. «Добежим вправо» до *, стоящей после дубля скопированной части исходного слова и на ее место запишем запомненную букву. Изменим состояние на q_6.

4. «Пробежим влево» вдоль дубля скопированной части исходного слова и на символе * изменим состояние на q_7.

5. «Пробежим влево» вдоль не скопированной части исходного слова до помеченного символа. Сдвинем головку на одну ячейку вправо и перейдем в состояние q_8.

6. Если обозревается буква, то перейдем в состояние q_1 и к пункту 1 нашего плана.

7. Если же обозреваемым символом является *, то это означает, что исходное слово полностью переписано, и перейдем в состояние q_9, сдвинув головку на одну ячейку влево.

8. В состоянии q_9 будем двигаться влево, каждый раз заменяя a на α, b на β, пока не увидим *.

9. Перейдем в заключительное состояние и сдвинем головку на одну ячейку вправо.

Для большей наглядности программы в тех случаях, когда не важно, какой именно из символов a или b записан в обозреваемой ячейке, будем в команде писать символ s. Аналогично вместо α или β будем писать символ t.

Пусть $s \in \{a, b\}$, $t \in \{\alpha, \beta\}$.

1.1) $q_1\ a \to q_2\ R\ \alpha$
1.2) $q_1\ b \to q_3\ R\ \beta$
2.1) $q_2\ s \to q_2\ R\ s$
2.2) $q_3\ s \to q_3\ R\ s$
2.3) $q_2\ * \to q_4\ R\ *$
2.4) $q_3\ * \to q_5\ R\ *$
3.1) $q_4\ s \to q_4\ R\ s$
3.2) $q_5\ s \to q_5\ R\ s$
3.3) $q_4\ * \to q_6\ a$
3.4) $q_5\ * \to q_6\ b$
4.1) $q_6\ s \to q_6\ L\ s$
4.2) $q_6\ * \to q_7\ L\ *$
5.1) $q_7\ s \to q_7\ L\ s$
5.2) $q_7\ t \to q_8\ R\ t$
6) $q_8\ s \to q_1\ s$
7) $q_8\ * \to q_9\ L\ *$
8.1) $q_9\ \alpha \to q_9\ L\ a$
8.2) $q_9\ \beta \to q_9\ L\ b$
9) $q_9\ * \to q_0\ R\ *$

Запишем протокол работы этой машины Тьюринга над словом ba.

$q_1\ ba$ (по 1.2) $\Rightarrow \beta\ q_3\ a$ (по 2.2) $\Rightarrow$
$\beta a\ q_3\ *$ (по 2.4) $\Rightarrow \beta a\ *q_5\ *$ (по 3.4) $\Rightarrow$
$\beta a\ *q_6\ b$ (по 4.1) $\Rightarrow \beta a\ q_6\ *b$ (по 4.2) $\Rightarrow$
$\beta\ q_7\ a*b$ (по 5.1) $\Rightarrow q_7\ \beta a*b$ (по 5.2) $\Rightarrow$
$\beta\ q_8\ a*b$ (по 6) $\Rightarrow \beta\ q_1\ a*b$ (по 1.1) $\Rightarrow$
$\beta\alpha\ q_2\ *b$ (по 2.3) $\Rightarrow \beta\alpha*\ q_4\ b$ (по 3.1) $\Rightarrow$
$\beta\alpha*b\ q_4\ *$ (по 3.3) $\Rightarrow \beta\alpha*b\ q_6\ a$ (по 4.1) $\Rightarrow$
$\beta\alpha*\ q_6\ ba$ (по 4.1) $\Rightarrow \beta\alpha\ q_6\ *ba$ (по 4.2) $\Rightarrow$
$\beta\ q_7\ \alpha ba$ (по 5.2) $\Rightarrow \beta\alpha\ q_8\ *ba$ (по 7) $\Rightarrow$
$\beta\ q_9\ \alpha*ba$ (по 8.1) $\Rightarrow q_9\ \beta a*ba$ (по 8.2) $\Rightarrow$
$q_9\ *ba*ba$ (по 9) $\Rightarrow q_0\ ba*ba$

Задание 2.

Написать машину Тьюринга, осуществляющую инверсию слова в алфавите A = {a, b}. (То есть записывающую последовательность букв, задающих слово, в обратном порядке. Например, инверсией слова АНЯ будет слово ЯНА)

Уровень 1. Определить внешний алфавит машины Тьюринга, исходную и заключительную конфигурации. Составить план работы машины Тьюринга.

Уровень 2. Написать программу машины Тьюринга и протоколы ее работы над словами *ba* и *babba*.

Уровень 3. Проведите исследования, сколько конфигураций в процессе работы этой машины Тьюринга появится при работе над словом, имеющим длину записи n.

МОДИФИКАЦИИ МАШИН ТЬЮРИНГА

Из приведенных примеров видно, что писать программы для машин Тьюринга достаточно трудоемко, и на первый взгляд совершенно непонятно, зачем же это нужно.

О том, зачем это нужно, подробно поговорим позже, когда будем рассматривать такие понятия, как сложность алгоритма (временная и ёмкостная). Сейчас только отметим, что все шаги машины Тьюринга считаются равными в смысле затрат времени на их выполнение.

Отметим основные положения, определяющие классическую машину Тьюринга. *Одна* лента. *Одна* головка. Команды согласованы.

Убирая эти ограничения, получим различные модификации машины Тьюринга.

МНОГОЛЕНТОЧНЫЕ МАШИНЫ ТЬЮРИНГА

Более точно, следовало бы написать *k*-ленточные машины Тьюринга при фиксированном *k*. В этой модели предполагается, что имеется *k* лент, на каждой из которых может быть записано свое слово. Головка обозревает одновременно по одной ячейке на каждой ленте и, в зависимости от их содержимого, может изменить или не изменять содержимое каждой из обозреваемых ячеек, сдвинуться (или не сдвигаться) на одну ячейку влево или вправо, причем на каждой ленте сдвиг может быть разным.

Команда k-ленточной машины Тьюринга имеет вид

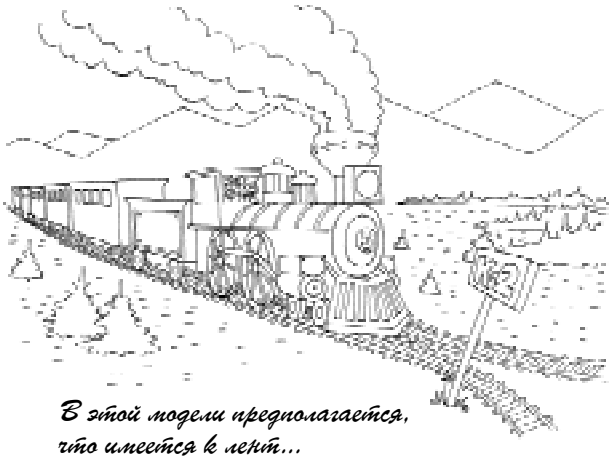
$$q_r\begin{pmatrix} a_i^1 \\ \vdots \\ a_i^k \end{pmatrix} \to q_t \begin{pmatrix} S^1 \\ \vdots \\ S^k \end{pmatrix}\begin{pmatrix} a_j^1 \\ \vdots \\ a_j^k \end{pmatrix}$$

где $S^1, \ldots, S^k \in \{L, R, \_\}$ и обозначают соответственно сдвиги влево, вправо или отсутствие сдвига головки.

Эта команда читается следующим образом: «Если машина Тьюринга находится в состоянии q_r и в обозреваемых ячейках лент записаны соответственно символы $a_i^1, \ldots, a_i^k$, то эти символы заменяются соответственно на $a_j^1, \ldots, a_j^k$, головка производит сдвиги $S^1, \ldots, S^k$ и машина Тьюринга переходит в состояние q_t».

На многоленточной машине Тьюринга программы для решения многих задач выглядят гораздо проще, чем соответствующие программы для одноленточной машины. Это связано с тем, что использование нескольких лент позволяет иметь одновременный доступ к нескольким (точнее, не более чем к *k*) различным записям.

Вернемся к задаче сложения двух чисел, записанных в двоичной системе счисления, для решения которой на одноленточной машине Тьюринга был разработан план



В этой модели предполагается, что имеется k лент...

программы, но сама программа не была приведена из-за ее чрезмерной громоздкости.

Пример 4.

Написать 3-ленточную машину Тьюринга, вычисляющую сумму двух чисел, записанных в двоичной системе счисления.

Начальная конфигурация машины Тьюринга будет $q_1\begin{pmatrix} X \\ Y \\ * \end{pmatrix}$, где X и Y – двоичные записи натуральных чисел. Требуется, чтобы заключительной конфигурацией была $q_0\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$, где Z – двоичная запись суммы. Для этого разработаем план работы машины Тьюринга.

1. Установим головку машины на последние символы записей чисел X и Y.

2. Будем складывать числа «в столбик» поразрядно,

3. Запоминая перенос единицы в следующий разряд с помощью дополнительного состояния.

4. Символ * перед более короткой записью числа будем интерпретировать как цифру 0.

5. При достижении левого края обеих записей остановиться.

$$1.1)\ q_1\begin{pmatrix} s_1 \\ s_2 \\ * \end{pmatrix} \to q_1 \begin{pmatrix} R \\ R \\ \end{pmatrix}\begin{pmatrix} s_1 \\ s_2 \\ * \end{pmatrix}$$

1.2) $q_1\begin{pmatrix}*\\ s_2\\ *\end{pmatrix}\to q_1\begin{pmatrix}R\end{pmatrix}\begin{pmatrix}*\\ s_2\\ *\end{pmatrix}$

1.3) $q_1\begin{pmatrix}s_1\\ *\\ *\end{pmatrix}\to q_1\begin{pmatrix}R\end{pmatrix}\begin{pmatrix}s_1\\ *\\ *\end{pmatrix}$

1.4) $q_1\begin{pmatrix}*\\ *\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ L\end{pmatrix}\begin{pmatrix}*\\ *\\ *\end{pmatrix}$

2.1) $q_2\begin{pmatrix}0\\ 0\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ L\\ L\end{pmatrix}\begin{pmatrix}0\\ 0\\ 0\end{pmatrix}$

2.2) $q_2\begin{pmatrix}0\\ 1\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ L\\ L\end{pmatrix}\begin{pmatrix}0\\ 1\\ 1\end{pmatrix}$

2.3) $q_2\begin{pmatrix}1\\ 0\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ L\\ L\end{pmatrix}\begin{pmatrix}1\\ 0\\ 1\end{pmatrix}$

2.4) $q_2\begin{pmatrix}1\\ 1\\ *\end{pmatrix}\to q_3\begin{pmatrix}L\\ L\\ L\end{pmatrix}\begin{pmatrix}1\\ 1\\ 0\end{pmatrix}$

3.1) $q_3\begin{pmatrix}0\\ 0\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ L\\ L\end{pmatrix}\begin{pmatrix}0\\ 0\\ 1\end{pmatrix}$

3.2) $q_3\begin{pmatrix}0\\ 1\\ *\end{pmatrix}\to q_3\begin{pmatrix}L\\ L\\ L\end{pmatrix}\begin{pmatrix}0\\ 1\\ 0\end{pmatrix}$

3.3) $q_3\begin{pmatrix}1\\ 0\\ *\end{pmatrix}\to q_3\begin{pmatrix}L\\ L\\ L\end{pmatrix}\begin{pmatrix}1\\ 0\\ 0\end{pmatrix}$

3.4) $q_3\begin{pmatrix}1\\ 1\\ *\end{pmatrix}\to q_3\begin{pmatrix}L\\ L\\ L\end{pmatrix}\begin{pmatrix}1\\ 1\\ 1\end{pmatrix}$

4.1) $q_2\begin{pmatrix}*\\ 0\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ L\end{pmatrix}\begin{pmatrix}*\\ 0\\ 0\end{pmatrix}$

4.2) $q_2\begin{pmatrix}*\\ 1\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ L\end{pmatrix}\begin{pmatrix}*\\ 1\\ 1\end{pmatrix}$

4.3) $q_2\begin{pmatrix}0\\ *\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ \\ L\end{pmatrix}\begin{pmatrix}0\\ *\\ 0\end{pmatrix}$

4.4) $q_2\begin{pmatrix}1\\ *\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ \\ L\end{pmatrix}\begin{pmatrix}1\\ *\\ 1\end{pmatrix}$

4.5) $q_3\begin{pmatrix}*\\ 0\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ L\end{pmatrix}\begin{pmatrix}*\\ 0\\ 0\end{pmatrix}$

4.6) $q_3\begin{pmatrix}*\\ 1\\ *\end{pmatrix}\to q_3\begin{pmatrix}L\\ L\end{pmatrix}\begin{pmatrix}*\\ 1\\ 0\end{pmatrix}$

4.7) $q_3\begin{pmatrix}0\\ *\\ *\end{pmatrix}\to q_2\begin{pmatrix}L\\ \\ L\end{pmatrix}\begin{pmatrix}0\\ \\ 0\end{pmatrix}$

4.8) $q_3\begin{pmatrix}1\\ *\\ *\end{pmatrix}\to q_3\begin{pmatrix}L\\ \\ L\end{pmatrix}\begin{pmatrix}1\\ *\\ 0\end{pmatrix}$

5.1) $q_2\begin{pmatrix}*\\ *\\ *\end{pmatrix}\to q_0\begin{pmatrix}R\\ R\\ R\end{pmatrix}\begin{pmatrix}*\\ *\\ *\end{pmatrix}$

5.2) $q_3\begin{pmatrix}*\\ *\\ *\end{pmatrix}\to q_0\begin{pmatrix}R\\ R\end{pmatrix}\begin{pmatrix}*\\ *\\ 1\end{pmatrix}$

Протокол работы этой трехленточной машины Тьюринга для конкретных исходных данных желающие могут написать в качестве упражнения.

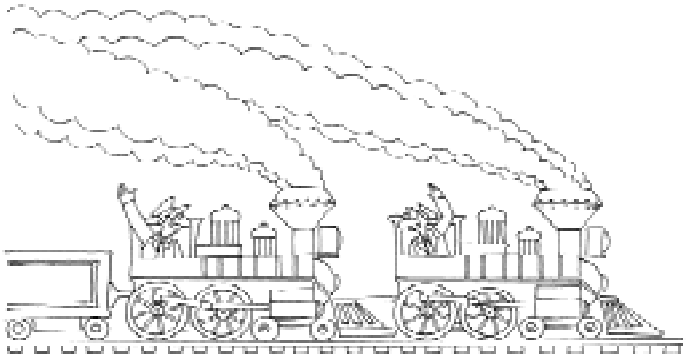
Задание 3.

Написать двуленточную машину Тьюринга, осуществляющую инверсию слова в алфавите $A = \{a, b\}$.

Уровень 1. Определить внешний алфавит машины Тьюринга, исходную и заключительную конфигурации. Составить план работы машины Тьюринга.

Уровень 2. Написать программу машины Тьюринга и протоколы ее работы над словами *ba* и *babba*.

Уровень 3. Проведите исследования, сколько конфигураций в процессе работы этой машины Тьюринга появится при работе над словом, имеющим длину записи *n*. Сравните полученное количество конфигураций с тем, которое получено в задании 2.



...имеется m головок, каждая из которых может обозревать одну ячейку.

МНОГОГОЛОВЧАТЫЕ МАШИНЫ ТЬЮРИНГА

Более точно, следовало бы написать *m*-головчатые машины Тьюринга при фиксированном *m*. В этой модели предполагается, что имеется m головок, каждая из которых может обозревать одну ячейку. В зависимости от содержимого всех обозреваемых ячеек машина может изменить или не изменять содержимое каждой из обозреваемых ячеек, сдвинуться (или не сдвигаться) на одну ячейку влево или вправо. В этой модели возможны две модификации: запрет на то, чтобы разные головки обозревали одну и ту же ячейку, либо головкам приписывается приоритет и в случае, если несколько головок обозревают одну и ту же ячейку, команда распространяется только на головку с наивысшим приоритетом, а остальные из этих головок пропускают свой шаг.

Команда m-головчатой машины Тьюринга имеет вид

$q_r\ (a_i^1, ..., a_i^m) \to q_t\ (S^1, ..., S^m)(a_j^1, ..., a_j^m)$,

где $S^1, ..., S^m \in \{L, R, \_\}$ и обозначают соответственно сдвиги влево, вправо или отсутствие сдвига головки.

Задание 4.

Написать двухголовчатую машину Тьюринга, осуществляющую инверсию слова в алфавите $A = \{a, b\}$.

Уровень 1. Определить внешний алфавит машины Тьюринга, исходную и заключительную конфигурации. Составить план работы машины Тьюринга.

Уровень 2. Написать программу машины Тьюринга и протоколы ее работы над словами *ba* и *babba*.

Уровень 3. Проведите исследования, сколько конфигураций в процессе работы этой машины Тьюринга появится при работе над словом, имеющим длину записи *n*. Сравните полученное количество конфигураций с тем, которое получено в заданиях 2 и 3.

Задание 5.

Как можно сделать обобщение машин Тьюринга с *k* лентами и *m* головками? Как при этом будет выглядеть команда?

Задание 6.

Тем, кто прочитал и разобрался в том, что написано выше, вероятно, уже понятно, что при выполнении действий «в столбик» удобно пользоваться многоленточными машинами Тьюринга с количеством лент, совпадающим с количеством записей, записанных «в столбик». Что делать, если количество таких записей меняется в зависимости от исходных данных (например, решаем системы с различным количеством уравнений)? Что, если сделать «ленту» плоской? Как при этом будет выглядеть команда?

НЕДЕТЕРМИНИРОВАННЫЕ МАШИНЫ ТЬЮРИНГА

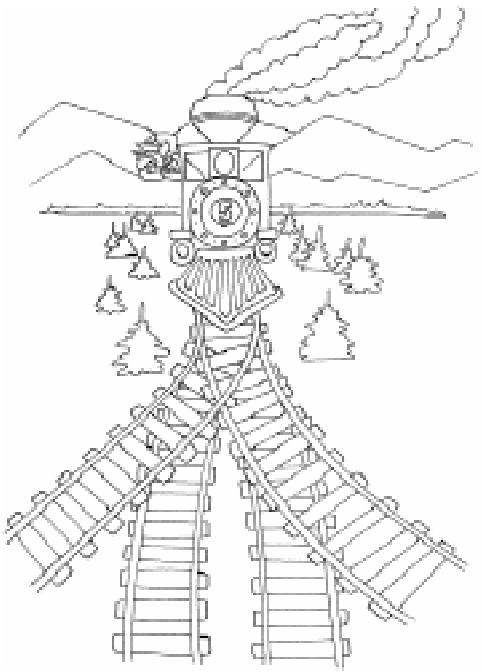
Недетерминированные машины Тьюринга получаются, если в программе клас-

сической машины Тьюринга разрешить использование несогласованных команд. Как же следует выполнить такую пару команд, если, например, одна предписывает изменить содержимое ячейки и в том же состоянии сдвинуться влево, а другая – не изменяя содержимого ячейки, сдвинуться вправо и изменить состояние? Для выполнения такой пары команд лента недетерминированной машины размножается, и на каждой ленте выполняется ровно одна из команд. Дальнейшие вычисления на каждой ленте продолжаются независимо.

Недетерминированные машины Тьюринга, как правило, используются для проверки истинности утверждений типа $\exists x\, P(x)$ (существует такой объект x, для которого справедливо утверждение $P(x)$). Для проверки истинности таких утверждений, как правило, работу недетерминированной машины Тьюринга разбивают на два этапа:

– этап угадывания, при реализации которого лента размножается, и на каждой из них выписывается «претендент» на решение;

– этап проверки, при реализации которого машина работает в детерминированном режиме и проверяет конкретного «претендента» на то, является ли он решением.



Недетерминированные машины Тьюринга получаются, если ... разрешить использование несогласованных команд...

Вместо одного заключительного состояния q_0 обычно используют два q_Y и q_N (Yes и No). Недетерминированная машина Тьюринга заканчивает работу в состоянии q_Y, если хоть на одной из лент она пришла в состояние q_Y. Если же на всех лентах недетерминированная машина Тьюринга пришла в состояние q_N, то и вся машина заканчивает работу в этом состоянии q_N.

ДЛЯ ЧЕГО ЖЕ НУЖНЫ МАШИНЫ ТЬЮРИНГА?

Прочитавший эту статью новичок-программист (считающий, что он съел собаку в программировании) может возразить, что все эти модификации – просто некоторые достаточно бедные и неудобные языки программирования. Мне, например, пришлось слышать от одного студента, что в течение многих лет он считал, что старички-программисты работают на машинах Тьюринга, а молодежь – на современных компьютерах.

Как уже было сказано, точные понятия алгоритма, в частности, машины Тьюринга были введены для доказательства несуществования алгоритма решения тех или иных задач. Однако именно развитие вычислительной техники стимулировало развитие такого направления в математике (и информатике), как теория сложности алгоритмов. Выяснилось, что для огромного класса задач, имеющих алгоритмы их решения, программы, реализующие эти алгоритмы для очень многих исходных данных, «зависают», то есть время их работы настолько велико, что приходится искать приближенные методы их решения, причем, чем больше точность решения задачи, тем дольше работает программа.

Машины Тьюринга оказались очень удобным математическим аппаратом, позволяющим получать оценки как времени реализации алгоритмов (в частности, и на реальных компьютерах), так и размера памяти, требуемой для вычислений.

В настоящее время активно исследуется вопрос о соотношении классов **P** и **NP**, определенных в терминах машин Тьюринга.

Класс **P** – это класс предикатов (то есть результатом их работы являются два значения – Да или Нет), вычислимых на машинах Тьюринга за число шагов, находящихся в полиноминальной зависимости от длины записи исходных данных.

Класс **NP** – это класс предикатов, вычислимых на недетерминированных машинах Тьюринга за число шагов, находящихся в полиноминальной зависимости от длины записи исходных данных.

Вопрос о том, совпадают ли эти два класса или имеется строгое включение $\mathbf{P} \subset \mathbf{NP}$, объявлен одной из сложнейших задач XXI века.

Несмотря на абстрактность этих определений, вопрос о совпадении или несовпадении этих двух классов означает, например, возможность или невозможность быстро (за полином шагов) решать любую переборную задачу (затратив минуты или часы, вместо годов или столетий, на ее решение).

Но об этом в следующий раз.

Косовская Татьяна Матвеевна, кандидат физико-математических наук, доцент кафедры математики Государственного Морского Технического Университета.