Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

(МАИ)

**Институт № 8 «Компьютерные науки и прикладная математика»**

**Курсовая работа**

**По фундаментальной информатике**

**1-й семестр**

**Выполнил:** Студент \_\_\_\_\_\_\_ Болотов Глеб Антонович

**Проверил:** \_\_\_\_\_\_\_ Чеснов Илья Игоревич

Москва, 2024

Содержание:

Введение 3

Теория 4-12

1. Курсовая работа 1: программирование машин Тьюринга 13-20

2. Курсовая работа 2: Диаграммы Тьюринга 21-28

3. Курсовая работа 3: Нормальные алгоритмы Маркова 29-34

Заключение 34

Список литературы 35

**Введение**

В данной курсовой работе мне необходимо построить алгоритмические модели на примере моделей Тьюринга и Маркова.

Цель: проиллюстрировать определения алгоритма путём построения алгоритмических моделей Тьюринга и Маркова.

Задачи:

1. составить программу машины Тьюринга в четвёрках, выполняющую заданное действие над словами, записанными на ленте;
2. разработать диаграмму Тьюринга решения задачи с использованием стандартных машин (r, l, R, L, K, a) и вспомогательных машин, определяемых задачей.
3. разработать нормальный алгоритм Маркова, обменивающий местами два троичных числа, разделённых знаком "^".
4. обобщить полученную информацию и сделать соответствующий вывод.

**Теория**

**Понятие алгоритмов**

Алгоритм — набор инструкций, описывающих порядок действий исполнителя для достижения результата решения задачи за конечное время. В старой трактовке вместо слова «порядок» использовалось слово «последовательность», но по мере развития параллельности в работе компьютеров слово «последовательность» стали заменять более общим словом «порядок». Это связано с тем, что работа каких-то инструкций алгоритма может быть зависима от других инструкций или результатов их работы. Таким образом, некоторые инструкции должны выполняться строго после завершения работы инструкций, от которых они зависят. Независимые инструкции или инструкции, ставшие независимыми из-за завершения работы инструкций, от которых они зависят, могут выполняться в произвольном порядке, параллельно или одновременно, если это позволяют используемые процессор и операционная система.

**Машины Тьюринга**

В 1936 г. аспирант Алан Тьюринг при исследовании алгоритмических проблем разрешимости (одной из знаменитых проблем Гильберта) предложил для уточнения понятия алгоритма использовать абстрактную вычислительную машину с очень простым набором операций. Построенная на базе этой машины алгоритмическая теория Тьюринга оказалась настолько плодотворной, что предвосхитила все последующие идеи универсальных вычислительных машин с программным управлением, включая идеи автоматизации программирования. В знак признания этих заслуг именем Тьюринга названа высшая награда АСМ, ежегодно присуждаемая за наиболее выдающиеся результаты в области информатики. Список лауреатов Тьюринговской премии можно найти на сайте АСМ http://www.acm.org/awards/taward.html. В 1942 г. группа британских математиков и филологов под руководством А. Тьюринга успешно применила теорию алгоритмов к разгадке шифров фашистской Германии. В послевоенное время Тьюринг принимал участие в создании реальных вычислительных машин, но на этот раз фортуна изменила ему и ни один из этих реальных проектов не был доведен до конца. Этому способствовали свойственные многим гениальным людям странности и чудачества в характере самого Тьюринга. Они же и привели к его трагической гибели от своеобразной английской рулетки: смешивая различные химические вещества он случайно получил яд и принял его вовнутрь. Недавно вышедшая книга Шарля Петцольда, посвящённая разбору основополагающего труда Тьюринга, ещё раз подтверждает актуальность Тьюринговской теории алгоритмов.

**Тезис Тьюринга-Черча**

Использование алгоритмических моделей опирается на тезис Тьюринга-Чёрча о том, что всякий интуитивный алгоритм может быть выражен средствами одной из алгоритмических моделей. Опровергнуть этот тезис пока никому не удалось. «Доказательством» же тезиса Тьюринга вполне может служить конструктивное построение фон Неймановской модели из Тьюринговской, предлагаемое в этом разделе курса.

Существенно, что все алгоритмические модели описывают один и тот же класс процессов обработки сообщений. Доказано, что одни модели сводятся к другим, т. е. всякий алгоритм, описанный средствами одной модели, может быть описан и средствами другой.

Наиболее близкой к вычислительным машинам является алгоритмическая модель Тьюринга. Поэтому именно эту модель мы выберем для формализации понятия алгоритма.

**Неформальное описание**

Машина Тьюринга состоит из ограниченной с одного конца бесконечной ленты, разделенной на ячейки, и комбинированной читающей и пишущей головки, которая может перемещаться вдоль ленты от ячейки к ячейке.

В каждой ячейке ленты может быть записан один знак алфавита А, называемого рабочим алфавитом МТ, либо пробел (его мы будем обозначать знаком λ).

Головка МТ в каждый момент времени располагается над одной из ячеек ленты, называемой рабочей ячейкой, и воспринимает знак, записанный в этой ячейке (букву алфавита А или Л). При этом головка находится в одном из конечного множества Q = {q0, q1, … qs} дискретных состояний, среди которых выделено одно начальное cостояние q0. В зависимости от состояния, в котором находится головка, и от буквы, записанной в рабочей ячейке, МТ выполняет одну из команд, составляющих ее программу.

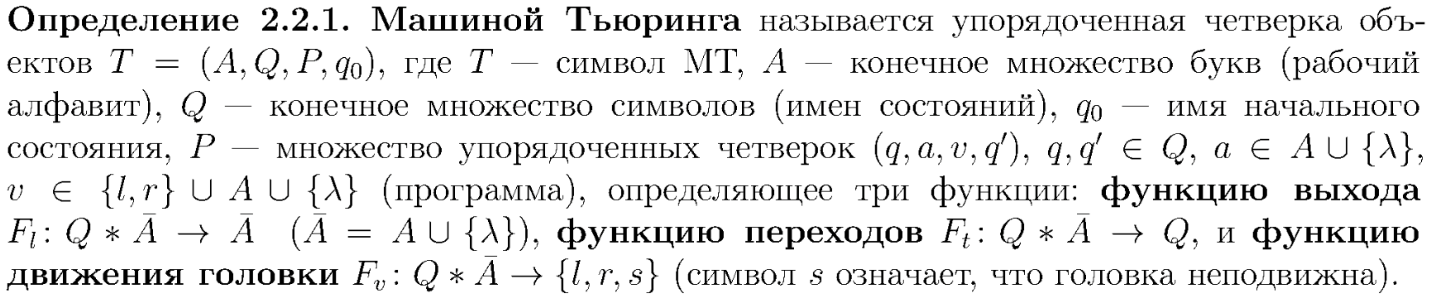
Выполнение команды состоит в выполнении элементарного действия, предписываемого этой командой, и переводе головки в новое состояние (которое, в частности, может совпадать со старым). Определено три вида элементарных действий: сдвиг головки на одну ячейку влево (если считать, что лента МТ ограничена слева, то для крайней левой ячейки сдвиг влево не определен), сдвиг головки на одну ячейку вправо, запись в рабочую ячейку какой-либо буквы рабочего алфавита А либо пробел (при этом буква, которая была записана в рабочей ячейке до выполнения записи, стирается). Таким образом, каждая команда может сменить рабочую ячейку, сделав новой рабочей ячейкой одну из соседних ячеек старой рабочей ячейки, или, не меняя рабочей ячейки, изменить знак, записанный в рабочей ячейке, и изменить состояние головки.

Перед началом работы МТ на ее ленту записывается исходное сообщение так, что в каждую ячейку ленты записывается одна буква сообщения, либо пробел. Будучи конечным, любое сообщение занимает конечное число ячеек ленты. При этом ячейки, расположенные справа от последней буквы исходного сообщения, заполняются пробелами и считаются пустыми. В начале работы МТ ее головка приводится в начальное состояние до и помещается над начальной рабочей ячейкой, которая определенным и фиксированным для каждой конкретной МТ образом расположена относительно исходного сообщения. Обычно в качестве рабочей ячейки мы будем брать ячейку, расположенную вслед за исходным сообщением, то есть, ячейку, содержащую символ пробела λ, расположенную непосредственно справа от последней буквы сообщения (все ячейки, расположенные справа от начальной рабочей ячейки, содержат в начале работы МТ символы λ). Пара (q0, а), где q0 - начальное состояние головки, а Є алфавиту A U {λ} - буква, записанная в начальной рабочей ячейке, определяет команду программы МТ, которая должна быть выполнена первой. В результате выполнения этой команды образуется новая пара (q, а'), где q - текущее состояние головки, а - буква, записанная в текущей рабочей ячейке, которая определяет следующую команду и т. д.

Таким образом, работа МТ полностью определяется ее программой и сообщением, которое было записано на ленте перед началом работы МТ.

Каждая команда МТ описывается упорядоченной четверкой символов (q, а, v, q'). Пара (q, а) определяет, какая из команд программы МТ должна быть выполнена (предполагается, что программа содержит не более одной команды, начинающейся с пары (q, a)).

**Математически сложное определение**

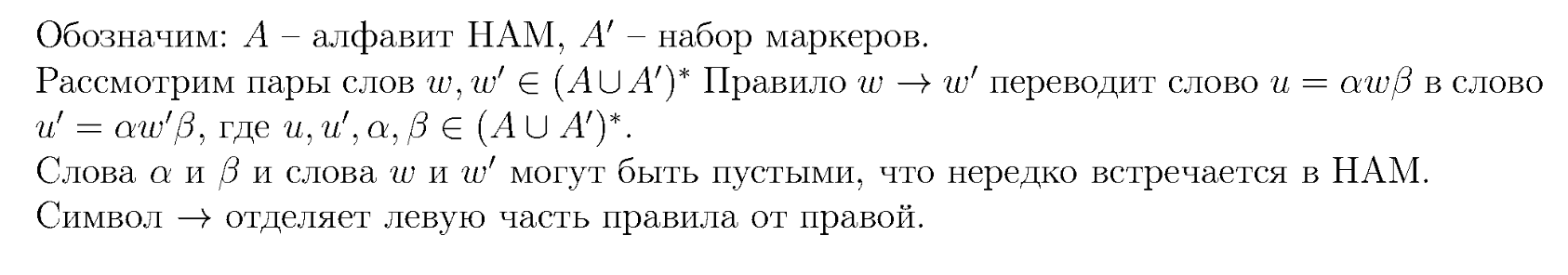
****

Замечание. Первоначально программы машин Тьюринга задавались наборами пятерок, включавших как функцию выхода, так и функцию движения. В частности, Шеннон доказывал свои знаменитые теоремы именно в такой форме записи. В пятерках движение головки и запись буквы дополняют друг друга, а в четверках они являются взаимно исключающими действиями. Кроме того, существуют и другие способы задания программ Тьюринга (диаграммы состояний, таблицы переходов).

**Нормальные алгоритмы Маркова**

В 1947–1954 гг. академик А. А. Марков (мл.) предложил алгоритмическую систему, в которой, как и в машине Тьюринга, преобразуются текстовые сообщения, но на основе других принципов (3, 25]. Элементарными тактами обработки алгоритмов Маркова являются замены подслов исходного сообщения на некоторые другие слова.

Алгоритмы Маркова представляют собой мощное средство описания, во многих случаях обладающее большей наглядностью и компактностью, например, по сравнению с машинами Тьюринга.

Нормальные алгоритмы Маркова по существу являются детерминистическими текстовыми заменами, которые для каждого входного слова однозначно задают вычисления и тем самым в случае их завершения порождают определенный результат. Это может быть обеспечено, например, установлением приоритета применения правил. Такие приоритеты могут быть заданы линейным порядком их записи 20|. В алгоритмической системе Маркова нет понятия ленты и подразумевается непосредственный доступ к различным частям преобразуемого слова.

Аналогично тезису Тьюринга-Черча, любой алгоритм в алфавите А может быть представлен нормальным алгоритмом Маркова над алфавитом А.

Примерно так же, как и для МТ, можно доказать алгоритмическую неразрешимость проблемы остановки и самоприменимости.

Существуют различные НАМ решения одной и той же задачи. Проблема построения алгоритма, который может определить эквивалентность любых двух НАМ, алгоритмически неразрешима.

Можно построить универсальный НАМ U, который мог бы интерпретировать любой нормальный алгоритм, включая самого себя.

**Диаграммы Тьюринга**

Диаграммы Тьюринга представляют одни МТ через другие, более простые МТ иным, визуально-топологическим способом, причём, как будет показано далее, этот способ не менее строг и полон, нежели "обычные" МТ. Так, машина, копирующая на ленте записанное на ней слово, рассмотренная в примере, может быть представлена через МТ , которые ищут начало слова на ленте, конец слова на ленте , копируют одну из букв слова и т . д. (имена состояний подобраны в этом примере так, чтобы легко было бы выделить более простые МТ ). Эти более простые МТ в свою очередь могут быть представлены через еще более простые МТ (это тоже нетрудно проиллюстрировать на конкретном примере. Такой нисходящий процесс представления МТ через более простые МТ должен обязательно оборваться, так как рано или поздно мы сведем описание каждой из рассматриваемых МТ к элементарным действиям, введенным при определении МТ.

При этом рассматриваемая МТ будет описана через элементарные МТ, т. е. такие, которые уже нельзя описать через более простые МТ , так как каждая из них выполняет всего одно элементарное действие останавливается.

**Курсовая работа 1: Программирование машин Тьюринга**

**Тема:** программирование машин Тьюринга

**Цель работы:** составить программу машины Тьюринга в четвёрках, выполняющую заданное действие над словами, записанными на ленте.

**Задание:** требуется написать алгоритм, который производит обмен разрядов двоичного числа, находящихся на четных и нечетных позициях.

**Средства, которыми буду пользоваться**: Машина Тьюринга в четвёрках, в эмуляторе jstu-4.2.3. Эмулятор имеет полубесконечную ленту и позволяет пошагово отлаживать программу (правила перехода).

**Идея и сценарий решения:**

Двигаемся справа налево по парам:

1) если первый символ равен 0:

- если второй символ равен 0, меняем его на !, двигаемся вправо и записываем 00;

- если второй символ равен 1, меняем его на !, двигаемся вправо и записываем 10;

2) если первый символ равен 1:

- если второй символ равен 0, меняем его на !, двигаемся вправо и записываем 01;

- если второй символ равен 1, меняем его на !, двигаемся вправо и записываем 11;

3) проделываем вышеописанные операции до лямбды, переворачиваем число справа и убираем появившиеся лишние пробелы;

4) программа успешно завершает работу;

**Правила перехода:**

*0, ,<,left1 - нулевое состояние*

*left1,1,<,viewnext1 - если число на вход - единица*

*left1,0,<,viewnext0 - если число на вход - ноль*

Далее приведен код, который идентичен по своей идеи выполнения и для входного 0, и для входной 1:

*viewnext1,1,!,truenext1*

*viewnext1,0,!,falsenext1*

*viewnext1, ,>,falsenext1*

*truenext1,!,>,pass11*

*pass11,1,>,pass11*

*pass11,0,>,pass11*

*pass11, ,>,copy11*

*copy11, ,!,put11*

*copy11,1,>,copy11*

*copy11,0,>,copy11*

*put11,!,1,put11*

*put11,1,>,put11*

*put11, ,1,nextstep1*

*falsenext1,!,>,pass10*

*falsenext1,1,>,pass10*

*falsenext1,0,>,pass10*

*pass10,1,>,pass10*

*pass10,0,>,pass10*

*pass10, ,>,copy10*

*copy10, ,!,put10*

*copy10,0,>,copy10*

*copy10,1,>,copy10*

*put10,!,0,put10*

*put10,0,>,put10*

*put10, ,1,nextstep0*

*nextstep1,0,<,nextstep1*

*nextstep1,1,<,nextstep1*

*nextstep1, ,<,bringback1*

*bringback1,1,<,bringback1*

*bringback1,0,<,bringback1*

*bringback1,!,1,start1*

*bringback1, , ,left1*

*nextstep0,0,<,nextstep0*

*nextstep0,1,<,nextstep0*

*nextstep0, ,<,bringback0*

*bringback0,0,<,bringback0*

*bringback0,1,<,bringback0*

*bringback0,!,0,start1*

*bringback0, , ,left1*

*start1,1,<,left1*

Первоначально программа воспроизводит ответ в перевернутом виде. Значит программе необходимо предоставить ответ в правильной виде. Предлагается копирование вправо вновь в перевёрнутым виде, но для этого необходимо поставить разделить между входными данными и перевернутым ответом, чтобы избежать затирания входных данных.

Также, необходимо убрать лишние пробелы, образовавшиеся после копирования ответа, для этого можно скопировать получившееся число влево до разделителя (который мы предварительно поставили перед переворачиванием ответа).

**Сложностная оценка решения:**

Сложность МТ равна O(n2), где n – длина ввода. Строго говоря, можно было бы избежать двух лишних проходов по бесконечной ленте МТ, но в представленной реализации это не учтено, так как приходится обрабатывать крайние значения (при входном нуле. Итого, в худшем случае программа пройдет по ленте МТ: 6 раз.

Выходит, что асимптотическая сложность алгоритма равна O(n2), так как все зависит от количества состояний по сути равных количеству проходов.

**Распечатка протокола и тесты:**

Тест 1

Входные данные: 1010

Выходные данные: 1010 101

Распечатка протокола:

1010

10!0 10

!010 1010

1010!101

1010 101 101

1010 101

1010 101

Тест 2

Входные данные: 111

Выходные данные: 111 1011

Распечатка протокола:

111

1!1 1

1!1 11  
…  
111 1101

111!1101

111!110& 1

…

111 1011

…

111 ! 101

111 1011

Тест 3

Входные данные: 0

Выходные данные: 0 0

Распечатка протокола:

0

0 0

0 00

0!00

0!

0&

0

0 0

**Дневник отладки**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб или дом | Дата | Время | Событие | Действие по исправлению | Примечание |
| 1 | Лаб | 30.09.2024 | 13:50 | Переписал 12 раз код | Познал дзен в МТ | Глупая ошибка |
| 2 | Дом | 30.10.2024 | 16:02 | Протестировал работу МТ с двумя 0, она ушла в бесконечность | Учёл этот случай в коде |  |
| 3 | Дом | 30.10.2024 | 16:45 | Реализовал удаление незначащих нолей в исходном и выходном числе | Добавил 12 новых состояний | Потратил много нервных клеток |

**Вывод:**

Подводя итог, хочется сказать, что это была моя первая встреча с МТ. Получил преогромное количество положительных эмоций и узнал много нового. Мне удалось реализовать даже больше мной задуманного, надеюсь приобретенные навыки пригодятся мне в дальнейшей учебе и работе.

**Курсовая работа 2: Диаграммы Тьюринга**

**Тема:** программирование машин Тьюринга при помощи диаграммера.

**Цель работы:** написать нормальный алгоритм Маркова, выполняющий заданное действие над словами.

**Задание:** требуется написать алгоритм, который выполняет логическое произведение (&& в С) двоичных чисел.

**Средства, которыми буду пользоваться**: диаграмма Тьюринга, в эмуляторе VirtualTuringMachine. Эмулятор имеет бесконечную ленту и позволяет пошагово отлаживать программу.

**Идея и сценарий решения:**

Сначала произвожу поразрядную конъюнкцию двух введенных чисел, результат получился перевернутым и вводные данные изменены: 0 и 1 заменены на ! на & соответственно. Далее проверяем является ли результат нулевым (0 или несколько 0 стоящих подряд), если результат нулевой стираем все ненужные нули (оставляю один 0) и возвращаю входные в их изначальное состояние. Иначе переворачиваю получившуюся ранее поразрядную конъюнкцию, удаляя при этом незначащие нули (каждый символ перевернутой поразрядной конъюнкции меняется на &). Далее удаляю все поставленные ранее & и сдвигаю результат программы к входнным данным так, чтобы их разделял пробельный символ. Возвращаю входные данные в их начальное состояние.

**Курсовая работа 3: Нормальные алгоритмы Маркова**

**Тема:** написание нормальных алгоритмов Маркова.

**Цель работы:** написать нормальный алгоритм Маркова, выполняющий заданное действие над словами.

**Задание:** входное слово представляет собой два троичных числа без знака, разделенные знаком ">". Составить алгоритм вычисления троичного логического сдвига первого числа вправо на число разрядов второго

**Средства, которыми буду пользоваться**: эмулятор нормальных алгоритмов Маркова, реализованный с помощью языка программирования python. Эмулятор открывает файл Markov.txt, в котором и описаны правила алгоритма.

**Идея и сценарий решения:**

В процессе обработки входных данных мы сначала доходим до конца второго троичного числа. Затем происходит удаление последнего символа этого числа. Далее мы перемещаемся влево, до конца первого числа, и в процессе этого перемещения удаляем первый символ, который идет после знака ">". После достижения конца первого числа мы размещаем ноль. После этого этапа вся процедура повторяется, причем количество повторений зависит от числа разрядов во втором входном числе.

**Правила перехода:**

\*0->0\*

\*1->1\*

\*2->2\*

>\*=>

\*>->>\*

1\*->x

2\*->x

0\*->x

1x->x1

2x->x2

0x->x0

0>x->x>

1>x->x>

2>x->x>

\*x->0\*

\*->

->\*

Мне достался довольно легкий вариант, особо интересных участков в программе я не нашел, поэтому решил выполнить еще один вариант для закрепления полученных навыков:

\*0->0\*

\*1->1\*

\*2->2\*

<\*=>

\*<-><\*

1\*->x

2\*->x

0\*->x

1x->x1

2x->x2

0x->x0

<x->0x<

\*x2->\*

\*x1->\*

\*x0->\*

\*->

->\*

**Сложностная оценка алгоритма:**

Сложность равна O(n), где n – длина ввода. Сложность линейная, так как асимптотика зависит от длины введенного слова.

**Распечатка протокола и тесты:**

Входные данные: 111>22

Выходные данные: 001

Распечатка протокола:

Тест 1

111>22

\*111>22

1\*111>22

...

111>22\*

111>2x

111>x2

11x>2

x11>2

\*011>2

...

001>

001

Входные данные: 22>1

Выходные данные: 02

Распечатка протокола:

Тест 2

\*22>1

2\*2>1

22\*>1

22>\*1

22>1\*

22>x

2x>

x2>

\*x2>

0\*2>

02\*>

02>\*

02

Входные данные: 01101>2221

Выходные данные: 00000

Распечатка протокола:

Тест 2

\*01101>2221

0\*1101>2221

01\*101>2221

011\*01>2221

0110\*1>2221

01101\*>2221

01101>\*2221

01101>2\*221

01101>22\*21

01101>222\*1

01101>2221\*

01101>222x

01101>22x2

01101>2x22

01101>x222

0110x>222

011x0>222

01x10>222

0x110>222

x0110>222

\*x0110>222

0\*0110>222

00\*110>222

001\*10>222

0011\*0>222

00110\*>222

00110>\*222

00110>2x2

…

00000

**Дневник отладки**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб или дом | Дата | Время | Событие | Действие по исправлению | Примечание |
| 1 | Дом | 15.10.2024 | 16:50 | Написал НАМ, забыл написать остановку | Написал остановку НАМ | Глупая ошибка |
| 2 | Дом | 15.10.2024 | 16:68 | Протестировал работу программы НАМ с 0 на конце, она ушла в бесконечность | Учёл этот случай в коде | Работает |
| 3 | Дом | 15.10.2024 | 18:34 | Написал программы для побитого сдвига влево, чтобы понять лучше | Добавил две операции, которые помогают выйти из рекурсии | Работает |

**Вывод:**

Это была моя первая встреча с нормальными алгоритмами Маркова. Мной был реализован алгоритм троичного логического сдвига вправо (и влево тоже) для входного троичного числа. Крайне приятная встреча после МТ.

**Заключение**

Итак, в этой курсовой работе я проиллюстрировал определения алгоритма путём построения алгоритмических моделей Тьюринга и Маркова. Как оказалось, это определение каждый раз отличалось.

В случае с Машиной Тьюринга алгоритмом назывались правила перехода, описанные для каждого возможного состояния и символа в текущей рабочей ячейке. Правила описывают действие, которое нужно совершить и состояние, в которое перейти.

В Диаграммах Тьюринга алгоритмом назывались схемы, состоящие из действий и стрелочек, обозначающих переходы.

В Нормальных Алгоритмах Маркова алгоритмом являются правила, которые описывают правила замены одних подстрок на другие подстроки. Для меня оказалось неожиданным, что на замене подстрок можно создать Тьюринг полный исполнитель.

Эти выводы я сделал в процессе выполнения задач по МТ, ДТ и НАМ, которые были успешно выполнены.

**Список литературы**

1. Гайсарян С. С., Зайцев В. Е., Курс информатики // Учебное пособие. – Изд-во Вузовская книга. – Москва, 2013. – 474 с.
2. Зайцев В. Е. и др., Информатика. Практикум. // Учебное пособие. – Москва, 1993.
3. Любимский Э. З., Мартынюк В. В., Элементы теории алгоритмов и структур данных. – МГУ. – Москва, 1976.
4. Шеннон К., Универсальная машина Тьюринга с двумя внутренними состояниями. – Работы по теории информации и кибернетике. – ИЛ. – Москва, 1963. – с. 740-750.
5. Лекции лауреатов премии Тьюринга. / Пер. с англ. – Мир. – Москва, 1993. – 560 с.
6. Марков А. А., Нагорный Н. М., Теория алгоритмов. – Наука. – Москва, 1984.