МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

КУРСОВАЯ РАБОТА

По дисциплине «Фундаментальная информатика»

На тему «Алгоритмические модели Тьюринга и Маркова»

Выполнил:
Студент группы М8О-108Б-23

Гаврилов Никита Валерьевич

Проверил:

Преподаватель

Севастьянов Виктор Сергеевич

Москва 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В рамках предмета фундаментальной информатики важно дать формальное определение алгоритма. Неформальное определение заключается в том, что алгоритм – это точно заданная последовательность правил, указывающая, каким образом можно за конечное число шагов получить выходное сообщение определенного вида, используя заданное входное сообщение, при этом, действия, предписываемые алгоритмом должны быть чисто механическими, всем понятными и легко выполнимыми. Данное определение слишком расплывчато, так как нет четкого определения слов «всем понятными и легко выполнимыми», порядок действий, кажущийся кому-то элементарным, может описываться весьма сложным алгоритмом. Кроме того, не все математические задачи алгоритмически разрешимы. При этом для установления неразрешимости какой-либо задачи необходимы суждения обо всех возможных алгоритмах, что неизбежно требует четкого определения того, что является алгоритмом, а что нет. Для этого и необходимо формальное и строгое определение алгоритма.

Формализация понятия алгоритма может быть произведена при помощи построения алгоритмических моделей. В данной работе будут рассмотрены два основных вида алгоритмических моделей: машины Тьюринга и нормальные алгоритмы Маркова. В первом случае алгоритм представляется в качестве процесса работы некоторой машины, способной выполнять небольшое число простых операций, во втором же алгоритм описывается, как набор преобразований слов в произвольных алфавитах.

Целью работы является иллюстрирование определения алгоритма путем построения алгоритмических моделей Тьюринга и Маркова.

В рамках работы будут выполнено составление алгоритмов для решения поставленных задач с помощью машины Тьюринга, эквивалентного ей графического представления – диаграммы Тьюринга, а также нормальных алгоритмов Маркова.

ГЛАВА 1. МАШИНЫ ТЬЮРИНГА

1.1 Вводная часть

Алгоритмическая модель машины Тьюринга была предложена английским математиком Аланом Тьюрингом в 1936 году.

Машина Тьюринга состоит из ограниченной с одного конца бесконечной ленты, разделенной на ячейки, а также комбинированной читающей и пишущей головки, которая может перемещаться вдоль ленты от ячейки к ячейке. В каждой такой ячейке может быть записан один знак некоторого алфавита, называемого рабочим алфавитом МТ, либо несобственный знак (пробел, обозначаемый λ). Головка МТ в каждый момент времени располагается над одной из ячеек ленты, называемой рабочей ячейкой, и воспринимает знак, записанный в этой ячейке. При этом головка находится в одном из конечного множества дискретных состояний, среди которых выделено одно начальное.

Согласно формальному определению, машиной Тьюринга называется упорядоченная четверка объектов $T = (A, Q, P, q0)$, где T – символ МТ, A – конечное множество букв (рабочий алфавит), Q – конечное множество символов (имен состояний), q0 – имя начального состояния, P – множество упорядоченных четверок (q, a, v, q'), $q, q' \in Q$, $a \in A \cup \{\lambda\}$, $v \in \{l, r\} \cup A \cup \{\lambda\}$ (программа), определяющее три функции: функцию выхода $Fl : Q \times \overline{A} \to \overline{A}(\overline{A} = A \cup \{\lambda\})$, функцию переходов $Ft : Q \times \overline{A} \to Q$, и функцию движения головки $Fv : Q \times \overline{A} \to \{l, r, s\}$ (символ s означает, что головка неподвижна).

Первоначально программы машин Тьюринга задавались наборами пятерок, включавших как функцию выхода, так и функцию движения. В пятерках движение головки и запись буквы дополняют друг друга, а в четверках они являются взаимно исключающими действиями.

Также целесообразным будет дать определение состоянию машины Тьюринга. Состоянием (или ситуацией) ленты МТ называется упорядоченная пара объектов $S = (z, k)$, где S – имя ситуации, z – сообщение, записанное на ленте, k – неотрицательное целое число, равное расстоянию (в ячейках) от края ленты до рабочей ячейки. Иными словами, k фиксирует положение рабочей ячейки на ленте. Иными словами, состояние – это все, что записано на ленте с выделенной рабочей ячейкой. Состояния записываются в наглядном виде, при котором левый край ленты обозначается квадратной

скобкой, правый, уходящий на бесконечность край – треугольной скобкой, а рабочая ячейка выделяется круглыми скобками (рисунок 1).

$$S = [a_{i_1} a_{i_2} \dots a_{i_{k-1}} (a_{i_k}) a_{i_{k+1}} \dots a_{i_n} \lambda\rangle$$

Рисунок 1 – Пример описания состояний МТ

Фактически, программа МТ задается упорядоченными наборами (в случае данной задачи - четверками) символов (командами) вида *a,b,c,d*, записанными каждая на отдельной строке, где *a* – имя состояния, в котором машина должна находиться в момент выполнения команды, *b* – знак, над которым должна располагаться головка машины в момент выполнения команды, *c* – знак, который нужно поставить на позицию, в которой в данный момент расположена головка, либо символ действия (*l, r, s*), которое нужно выполнить, *d* – имя состояния, в которое машина должна перейти после выполнения команды. Команды выполняются путем сопоставления конечных и начальных состояний, а также знаком, расположенным в текущей ячейки на ленте МТ, и порядок их выполнения не зависит от порядка, в котором они записаны в тексте программы.

В рамках решения поставленной задачи будет использоваться интерпретатор программ машины Тьюринга в четверках *jstu4*, реализованный в формате web-страницы (рисунок 2), принимающий на ввод текст программы и входное сообщение и выдающий на выходе преобразованное сообщение.

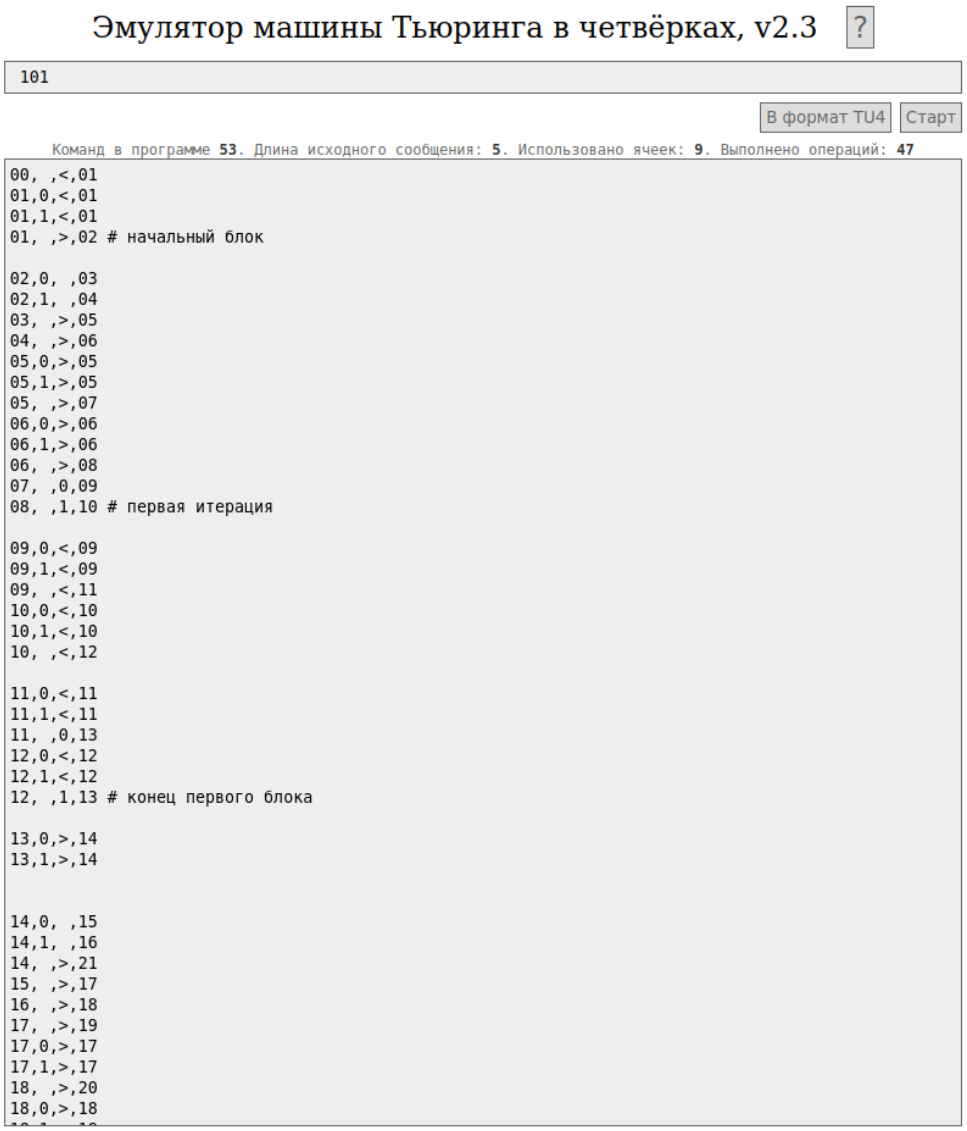

Рисунок 2 – Интерфейс интерпретатора *jstu4*

Интерфейс интерпретатора также допускает использование комментариев.

1.2 Постановка задачи

В рамках задачи, выданной под вариантом №3, необходимо составить алгоритм, меняющий два числа местами. Числа на вход интерпретатора поступают, разделенные пробелом, перед первым числом также расположен пробел. Рабочим алфавитом машины будет являться множество {0, 1}.

1.3 Идея решения задачи

Идея решения задачи заключается в следующих шагах:

Копирование второго числа: Создать копию второго числа, разместив её справа от исходных чисел. Это обеспечит нормированность исходных данных.

Копирование первого числа: Скопировать первое число после скопированного второго.

1.4 Описание алгоритма

В рамках первого этапа будет производиться копирование второго числа в область, отделенную от его правого края одним пробелом. В начальном состоянии головка находится непосредственно справа от второго числа (S_1). В процессе копирования головка МТ будет пробегать все второе число слева направо (S_2), затем смещаться на слово влево, копировать первое число и помещать его справа от скопированного второго числа (S_3), Далее головка МТ будет двигаться вправо, пока не дойдет до правой границы скопированного первого числа и остановится справа от него (S_4).

$$S_1 = [\lambda 1101 \lambda 1011 (\lambda)>$$
$$S_2 = [\lambda 1101 (\lambda) 1011 \lambda 1011>$$
$$S_3 = [(\lambda) 1101 \lambda 1011 \lambda 1011 \lambda 1101>$$
$$S_4 = [\lambda 1101 \lambda 1011 \lambda 1011 \lambda 1101 (\lambda)>$$

Рисунок 3 – Некоторые рассмотренные состояния МТ

Исходный код программы представлен в приложении А.

1.5 Тестирование и оценка сложности

В рамках тестирования работоспособности алгоритма на ввод программы были переданы несколько чисел краевого вида, такие как (0 0), (1 1), (000 111) и другие. Полный список рассмотренных тестовых случаев представлен в таблице 1.

Таблица 1 – Список рассмотренных тестовых случаев

Ввод	Вывод
0 0	0 0
0 1	1 0
0000 0000	0000 0000
1101 1011	1011 1101
111 000	111 000
1010101 010	010 1010101
11011101 110110	110110 1011101

Сложность алгоритма оценивалась по количеству элементарных операций (команд), выполненных машиной во время выполнения алгоритма. Эта величина зависит только от длины входного сообщения. Результаты тестирования на входных сообщениях разной длины приведены в таблице 2.

Таблица 2 – Зависимость входного сообщения и количества операций

Длина входного сообщения	Количество выполненных операций
2	88
4	228
8	700
16	2412
32	8908

На основании вышеприведенных результатов можно отметить, что при увеличении длины входного сообщения в 2 раза количество выполненных операций увеличивается больше чем в 2, но меньше чем в 4 раза, из чего можно сделать вывод, что сложность алгоритма приблизительно равна O($n$log$n$).

1.6 Выводы по главе

Разработан алгоритм копирования чисел, который меняет местами два числа, составлена соответствующая программа Машины Тьюринга в четверках. Получен практический опыт работы с абстрактным исполнителем Машина Тьюринга, написания алгоритма обработки двоичных чисел на языке предельно низкого уровня.

Получен опыт составления алгоритмов на языке низкого уровня, при котором необходимо многие элементарные действия (перемещение головки по слову, копирование знака и т. д.) задавать большим количеством команд. Вызвано это, прежде всего, тем, что единственным способом «запоминания» какого либо знака или выполнения какого-либо действия является ассоциация его с конкретным состоянием или конкретным знаком. Поэтому, для выполнения каждого элементарного действия нужно задавать столько состояний, над сколькими знаками может оказаться головка машины в момент выполнения этого действия, и для каждого состояния дублировать однотипные команды. Поэтому, при наличии более объемного алфавита, чем в условии выполненной задачи, код программы был бы гораздо более громоздким.

ГЛАВА 2. ДИАГРАММЫ ТЬЮРИНГА

2.1 Вводная часть

Диаграммы Тьюринга являют собой графическое представление машины Тьюринга, они позволяют избавиться от нагромождения состояний, упомянутого в качестве недостатка в выводах к предыдущей главе. Фактически, диаграммы Тьюринга представляют одни МТ через другие, более простые МТ иным, визуально-топологическим способом. В общем случае, диаграммы инкапсулируют основные действия, такие, как перемещение головки по слову и копирование слов в готовые элементарные машины, которые выполняют эти действия без ручной ассоциации с конкретными знаками, над которыми это действие выполняется. Кроме того, имеется вводить в процессе разработки алгоритма собственные машины, выделяя в них часто повторяющиеся действия, тем самым разделяя тело алгоритма на отдельные структуры и дополнительно сокращая его объем.

Состояния в диаграммах обозначаются точками, символ каждой подмашины в диаграмме ограничен слева и справа точками, обозначающими имя текущего состояния и состояния, в которое переходит машина после выполнения действия подмашины, соответственно. Если после выполнения действия одной подмашины, необходимо выполнить действие второй подмашины, то правая точка первой машины соединяется с левой точки второй машины, а над стрелкой указываются знаки, над которыми должна находиться головка машины, чтобы выполнить действие подмашины, к которой ведет стрелка. Если выполнение действия подразумевается для всех знаков рабочего алфавита, то над стрелкой ничего не указывается.

Для повторения какого-либо участка диаграммы до тех пор, пока в рабочей ячейке находится одна из букв некоторого фиксированного набора, необходимо замкнуть этот участок диаграммы стрелкой с соответствующей надписью. Прекращение повторения осуществляется по букве, не входящей в этот набор.

Таким образом, диаграмма Тьюринга состоит из символов (имен) машин Тьюринга, точек и стрелок, над которыми написаны знаки рабочего алфавита.

При составлении алгоритмов с помощью диаграмм Тьюринга, диаграмму можно составить, как изобразив на бумаге, так и с помощью специального программного обеспечения, называемого диаграммером, который обеспечивает интерактивное составление диаграммы и исполнение описываемого ею алгоритма. В рамках выполнения нижеследующего задания будет использоваться диаграммер *JDT Version 2.1* (рисунок 4).

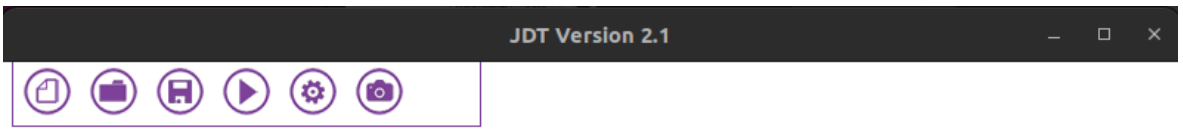

Рисунок 4 – Интерфейс диаграммера *JDT Version 2.1*

2.2 Постановка задачи

В рамках поставленной задачи, выданной под вариантом №32, необходимо сконструировать диаграмму Тьюринга, реализующую алгоритм, который прибавляет единицу к шестнадцатеричному числу.

2.3 Идея решения задачи

В первую очередь реализуемый диаграммой алгоритм должен скопировать введенное число.

После копирования головка будет проходить копию числа справа налево, применяя подмашину, которая прибавляет единицу к цифре.

Если после применения остается 0, тогда головка перемещается на ячейку влево и снова применяет подмашину, которая прибавляет единицу к цифре.

После прибавления головка перемещается к началу скопированного числа и применяет подмашину, которая убирает незначащие нули.

После того, как подмашина избавилась от незначащих нулей, применяется подмашина, которая перемещает слово влево, потом головка перемещается справа от скопированного числа и алгоритм заканчивается.

2.4 Описание алгоритма

Диаграмма основной машины представлена на рисунке 5.

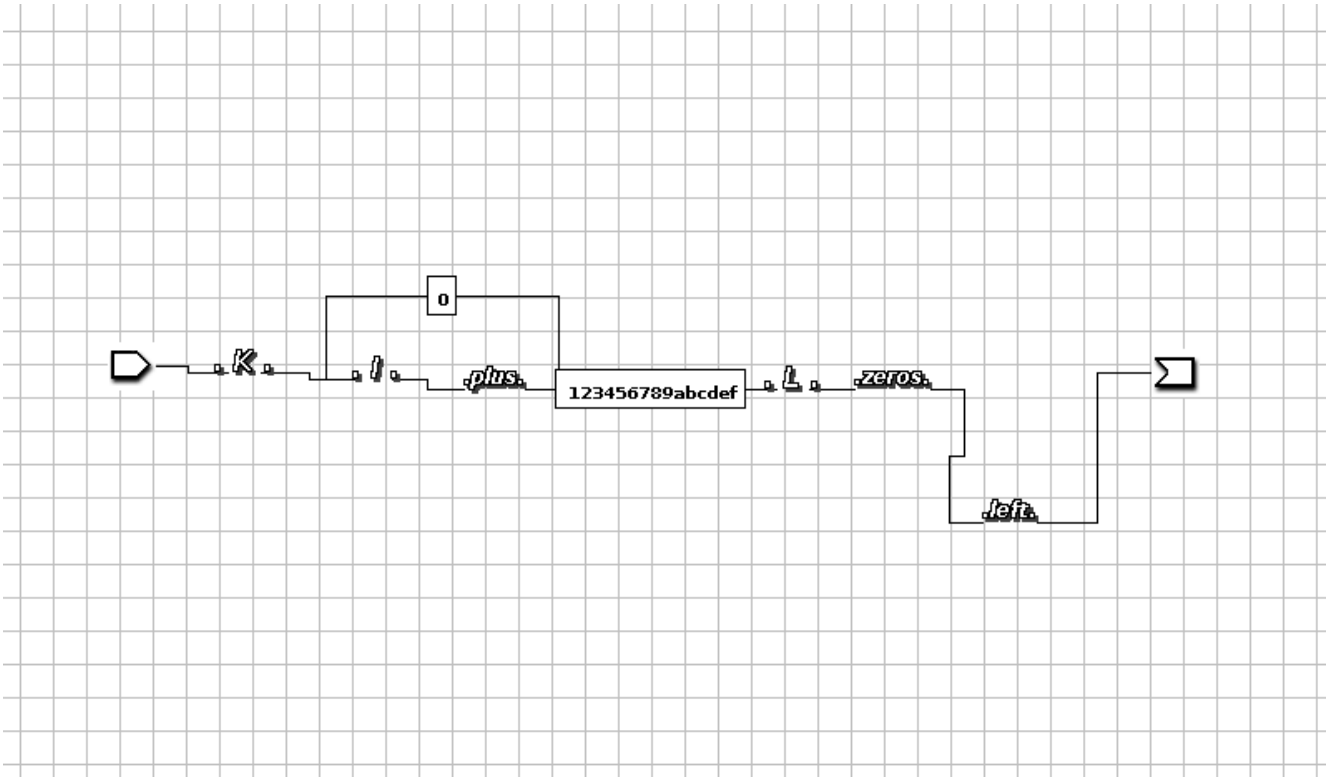

Рисунок 5 – Диаграмма основной машины

Машина прибавления *PLUS* (рисунок 6) В зависимости от символа выполняет определенные действия. Если это цифра, то прибавляется единица, f заменяется на 0, т. к. f - максимальная цифра 16-ричной системы. Если же это пробел, то головка сдвигается вправо.

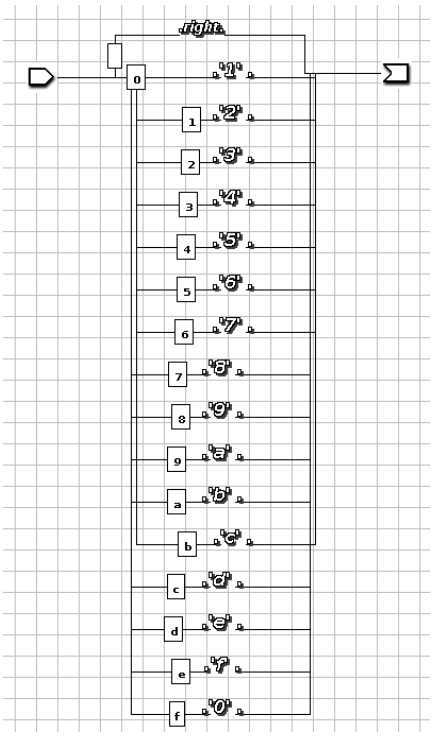

Рисунок 6 – Машина *PLUS*

Далее если в ячейке ноль машина *PLUS* повторяется, иначе головка перемещается на слово влево (слева от скопированного числа).

После завершения описанной выше обработки запускается машина ZEROS (рисунок 7), которая затирает незначащие нули.

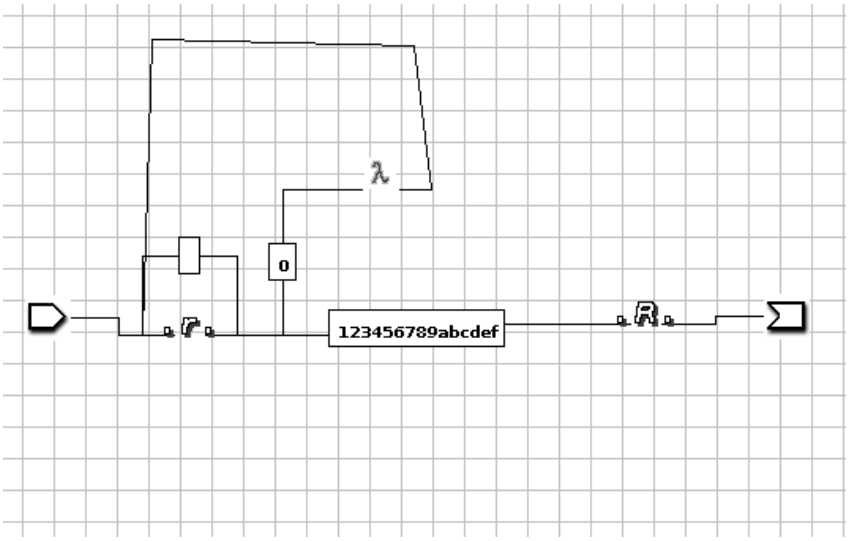

Рисунок 7 – Машина *ZEROS*

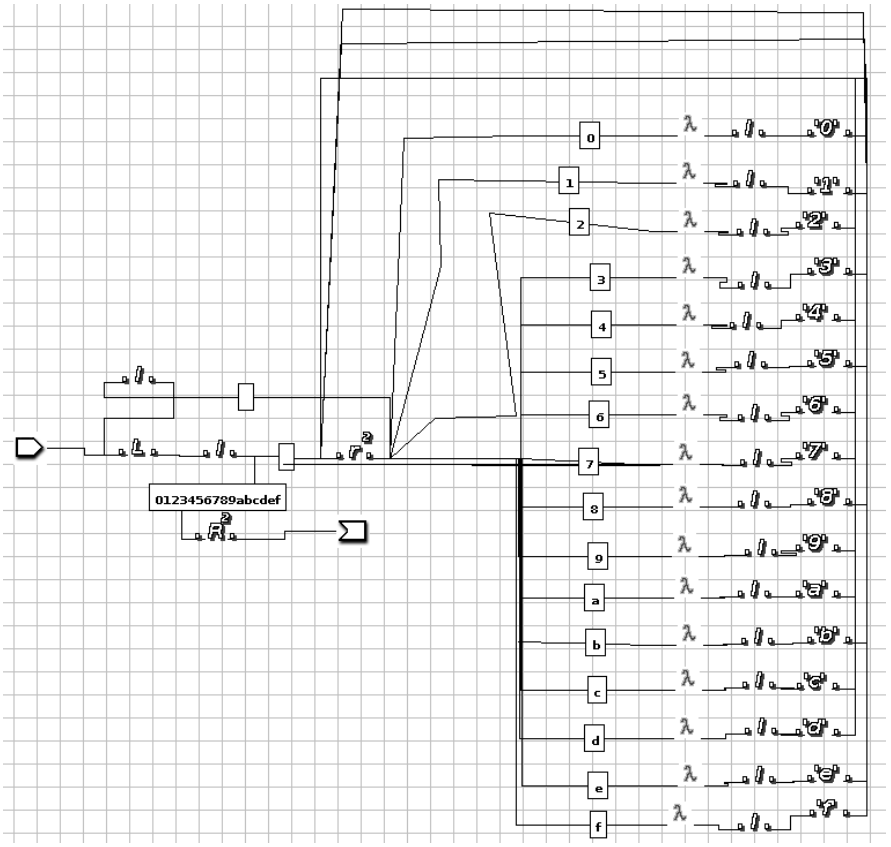

Рисунок 8 – Машина *LEFT*

После этого применяется машина *LEFT* (рисунок 8). Предназначенная для сдвига результата влево, до тех пор, пока между исходным числом и результатом не останется один пробел. Далее головка ставится справа от результата, на чем выполнение алгоритма завершается.

2.5 Тестирование и оценка сложности

В рамках тестирования алгоритма на вход диаграмме были представлены несколько строк краевого вида (одни нули, одна цифра, число с ведущими нулями), а также числа с четным и нечетным числом цифр. Полный список рассмотренных тестовых случаев представлен в таблице 4.

Таблица 4 – Список рассмотренных тестовых случаев

Входные данные	Выходные данные	Описание тестируемого случая
0000	1	Число из одних нулей
001	2	Единица с незначащими нулями
ffff	10000	Особенный случай
10fabfff	10fac000	Большое число

Так как использованный интерпретатор не приводит данных о количестве выполненных операций, оценка сложности производилось с засечением времени выполнения программы для каждого тестового случая. В таблице 5 приведена зависимость времени выполнения алгоритма от объема входного сообщения.

Таблица 5 – Зависимость времени выполнения алгоритма от размера сообщения

Длина входного сообщения	Примерное время выполнения, с.
4	0.5
8	0.98
16	1.36
32	2.03

На основании вышеприведенных данных можно утверждать, что при увеличении объема входного сообщения в 2 раза время выполнения также возрастает приблизительно в 2 раза, значит, в соответствии с заданием, сложность алгоритма линейна и равна O(n).

2.6 Выводы по главе

Разработан и реализован в среде разработки Диаграмм Тьюринга алгоритм, считающий инкремент для шестнадцатеричного числа. На примере работы с более высокоуровневой реализацией абстрактного исполнителя Машины Тьюринга дополнен опыт разработки алгоритмов, полученный при решении предыдущей задачи.

По сравнению со средой Машины Тьюринга, при работе с диаграммой удобнее задавать в алгоритме рутинные операции, не задумываясь о наименовании состояний и т. п., а также создавать дополнительные машины, выделяя в них объемные, часто повторяющиеся операции. С другой стороны, громоздкий интерфейс диаграммера по сравнению с командной строкой интерпретатора Машины Тьюринга не слишком удобен для быстрой отладки алгоритма.

С другой стороны, диаграммы в некоторой степени удобны для написания их без использования интерпретатора (на бумаге) и последующей отладки путем ручного выполнения (проговаривания) алгоритма.

ГЛАВА 3. НОРМАЛЬНЫЕ АЛГОРИТМЫ МАРКОВА

3.1 Вводная часть

Рассматриваемая алгоритмическая система была предложена академиком А. А. Марковым в 1947-1954 гг. Аналогично машине Тьюринга, в этой модели происходит преобразование текстовых сообщений, основанное на замене подслов исходного сообщения на некоторые другие слова.

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. В алгоритмической системе Маркова нет понятия ленты, и подразумевается непосредственный доступ к различным частям преобразуемого слова.

В качестве основных принципов выполнения алгоритма в этой системе можно выделить следующие:

1. Если применимо несколько правил, то берется правило, которое встречается в описании алгоритма первым;
2. Если правило применимо в нескольких местах обрабатываемого слова, то выбирается самое левое из этих мест.

Таким образом, нормальный алгоритм Маркова (НАМ) представляет собой упорядоченный набор правил-продукций – пар слов (цепочек знаков, в том числе пустых цепочек длины 0), соединенных между собой символами $\rightarrow$ или $\mapsto$. Каждая продукция представляет собой формулу замены части входного слова, совпадающей с левой частью формулы, на ее правую часть.

Процесс выполнения НАМ заканчивается в одном из двух случаев: либо все формулы оказались неприменимыми, то есть в обрабатываемом слове нет вхождений левой части ни одной формулы подстановки; либо только что применилась так называемая терминальная (завершающая) продукция, в которой правую и левую часть разделяет символ ↦. Терминальных продукций в одном НАМ может быть несколько.

В любом из этих случаев НАМ применим к данному входному слову. Если в процессе выполнения НАМ бесконечно долго применяются нетерминальные правила, то алгоритм неприменим к данному входному слову. Существуют следующие достаточные признаки применимости НАМ ко всем входным словам:

1. Левые части всех продукций непустые, а в правых частях нет букв, входящих в левые части;
2. В каждом правиле правая часть короче левой части.

При составлении НАМ был использован интерпретатор представленный на рисунке 10.

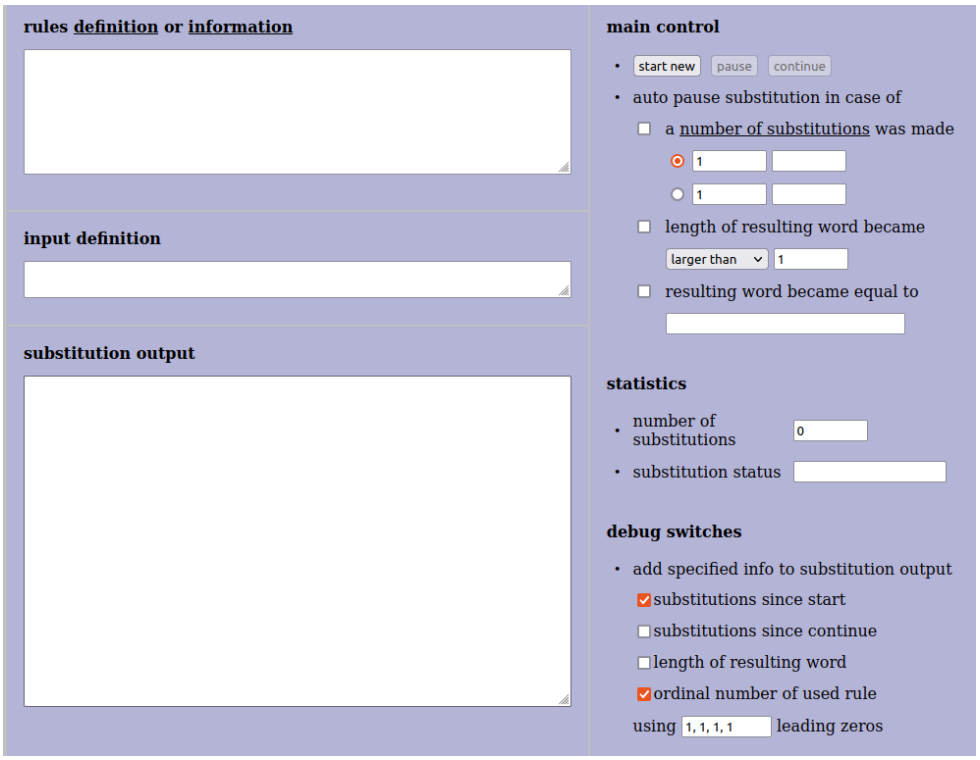

Рисунок 10 – Интерфейс интерпретатора *markov*

В среде данного интерпретатора вместо символа ↦ для обозначения терминальной продукции используется точка, поставленная сразу после правой части формулы. Если точка используется как непосредственно слово для подстановки, то она выделяется одинарными кавычками.

Также в качестве заметной особенности модели НАМ можно отметить, что здесь, в отличие от моделей Тьюринга, вывод не обязан быть нормированным, то есть входные данные не должны оставаться в неизменном виде после завершения программы. Связано это со сложностью организации нормированного вывода, а также с отсутствием перемещающейся линейно рабочей головки.

3.2 Постановка задачи

В рамках поставленной задачи, выданной под вариантом №40 необходимо составить алгоритм вычисления двоичного числа - двоичного логарифма двоичного числа.

3.3 Идея решения задачи

Идея решения заключается в том, чтобы посчитать длину исходного числа без одного символа в двоичной системе счисления. То есть будет счетчик в двоичной системе, который будет забирать по цифре из исходного числа и прибавлять к счетчику, пока исходное число не пропадет полностью.

3.4 Описание алгоритма

Сначала перемещаем указатель (символ) в конец числа и затираем цифру, далее забираем правую цифру указателем, перемещаем вправо и прибавляем к счетчику. Перемещаемся к следующей цифре и повторяем те же действия. В конце, когда числа не осталось указатель не находит ничего и заменяет себя на другой символ, который удаляет себя и заканчивает алгоритм.

Исходный код алгоритма представлен на рисунке 11.

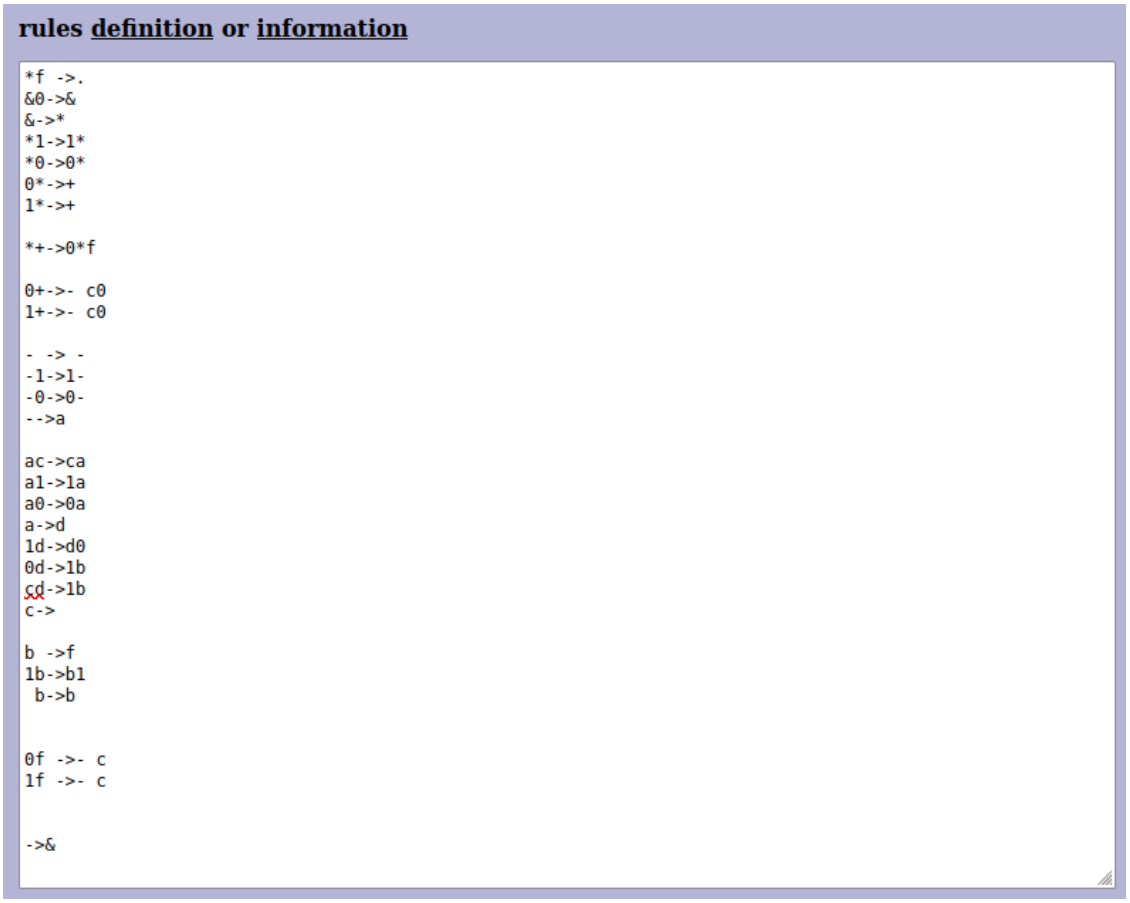

Рисунок 11 – Исходный код алгоритма

3.5 Тестирование

В рамках тестирования алгоритма на вход программе были представлены несколько чисел различного вида. Полный список рассмотренных тестовых случаев представлен в таблице 7.

Таблица 7 – Список рассмотренных тестовых случаев

Входные данные	Выходные данные	Описание тестируемого случая
1	0	2^0 = 1
10	1	2^1 = 2
110	10	2^2 = 4 (4<110<8) округление в меньшую сторону
1010	11	2^3 = 8 (8<1010<16) округление в меньшую сторону
10101	100	2^4 = 16 (16<10101<32) округление в меньшую сторону

Для проведения оценки сложности, по аналогии с предыдущим заданием, использовалось количество итераций алгоритма для входных сообщений различной длины. Результаты наблюдений приведены в таблице 8.

Таблица 8 – Зависимость количества итераций алгоритма от размера сообщения

Длина входного сообщения	Количество итераций
1	8
2	19
4	45

На основании вышеприведенных данных можно утверждать, что при увеличении объема входного сообщения в 2 раза время выполнения тоже возрастает примерно в 2 раза, следовательно, сложность алгоритма линейна и равна O(*n*).

3.6 Выводы по главе

Разработан и реализован в среде разработки алгоритмов алгоритмической модели Маркова алгоритм вычисления двоичного числа - двоичного логарифма двоичного числа. Дополнен опыт решения предыдущих задач в области создания алгоритмов в различных алгоритмических моделях.

В отличие от предыдущих моделей, нормальные алгоритмы Маркова сложнее по свой структуре и в процессе реализации, так как отсутствует строгий контроль над текущим состоянием исполнителя (положением некоторой головки), не зависящий от фактического содержания обрабатываемых данных. Кроме того, во время разработки необходимо отслеживать порядок выполнения правил в зависимости от их местоположения в исходном коде, каковая особенность отсутствовала в моделях Тьюринга. В целом, решение данной задачи не вызвало вышеописанных трудностей, в связи с простотой ее условия.

ЗАКЛЮЧЕНИЕ

Итак, в рамках выполнения данной курсовой работы были проиллюстрированы определения алгоритма при помощи алгоритмических моделей Тьюринга и Маркова. С этой целью были выполнены предложенные задания с использованием среды машины Тьюринга, диаграмм Тьюринга и нормальных алгоритмов Маркова. Каждая из рассмотренных алгоритмических моделей имела свои особенности.

Модель машины Тьюринга отличалась особенной низкоуровневостью и строгостью выполнения алгоритма. Было необходимо обеспечить нормированный ввод, при котором входное сообщения должно оставаться в неизменном виде, а каждое действие, совершаемое алгоритмом (рабочей головкой) зависело исключительно от его местоположения на ленте и значения стоящего в этом месте знака. Из-за низкоуровневости модели код ее алгоритма был громоздким и весьма объемным, но при этом подробное описание каждого действия позволяло лучше контролировать совершаемые машиной действия и повышало точность ее работы.

С другой стороны, диаграммы Тьюринга полностью эквивалентны машине, и лишь графически представляет ее алгоритм, инкапсулируя при этом рутинные и элементарные операции в отдельные подмашины. Таким образом, ускоряется и упрощается разработка и отладка алгоритма. При этом, громоздкий интерфейс интерпретатора, нагромождения соединяющих подмашины стрелок значительно снижают заметность этого преимущества.

Наконец, нормальные алгоритмы Маркова значительно отличались от предыдущих моделей, так как их выполнение зависело не от пространственного положения рабочей головки (состояния машины), а от фактического содержания входного сообщения и взаимного расположения его знаков. Данная особенность несколько усложняла разработку алгоритма, так как его поведение могло значительно меняться в зависимости от содержания сообщения. Так же усложнялись пространственно зависимые операции, такие как копирование слов. В том числе поэтому, при разработке алгоритма в системе Маркова не нужно было обеспечивать нормированный вывод.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гайсарян С. С., Зайцев В. Е., Курс информатики // Учебное пособие. – Изд-во Вузовская книга. – Москва, 2013. – 474 с.

2. Г.-Д. Эббинхауз, К. Якобе, Ф.-К. Ман, Г. Хермес, Машины Тьюринга и рекурсивные функции. – Мир. – Москва, 1972.

3. Любимский Э. З., Мартынюк В. В., Элементы теории алгоритмов и структур данных. – МГУ. – Москва, 1976.

4. Бауэр Ф., Гооз Т., Информатика. – Мир. – Москва, 1976, 1990.

5. Лекции лауреатов премии Тьюринга. / Пер. с англ. – Мир. – Москва, 1993. – 560 с.

6. Марков А. А., Нагорный Н. М., Теория алгорифмов. – Наука. – Москва, 1984.

7. Зайцев В. Е. и др., Информатика. Практикум. // Учебное пособие. – Москва, 1993.

8. Шеннон К., Универсальная машина Тьюринга с двумя внутренними состояниями. – Работы по теории информации и кибернетие. – ИЛ. – Москва, 1963. – с. 740-750.

ПРИЛОЖЕНИЕ А

Исходный код алгоритма решения задачи для машины Тьюринга

Далее: ВЧ – второе число, ПЧ – первое число

Копирование второго числа

1. Переход в начало ВЧ

```
00, ,<,01
01,0,<,01
01,1,<,01
01, ,>,02
```

2. Процесс копирование ВЧ

```
02,0, ,03
02,1, ,04
03, ,>,05
04, ,>,06
05,0,>,05
05,1,>,05
05, ,>,07
06,0,>,06
06,1,>,06
06, ,>,08
07, ,0,09
08, ,1,10
09,0,<,09
09,1,<,09
09, ,<,11
10,0,<,10
10,1,<,10
10, ,<,12
```

```
11,0,<,11
11,1,<,11
11, ,0,13
12,0,<,12
12,1,<,12
12, ,1,13
13,0,>,14
13,1,>,14
14,0, ,15
14,1, ,16
14, ,>,21
15, ,>,17
16, ,>,18
17, ,>,19
17,0,>,17
17,1,>,17
18, ,>,20
18,0,>,18
18,1,>,18
19,0,>,19
19,1,>,19
19, ,0,09
20,0,>,20
20,1,>,20
20, ,1,10
```

3. Переход к ПЧ

```
21,0,<,21
```

```
21,1,<,21
21, ,<,22
22,0,<,22
22,1,<,22
22, ,<,23
23,0,<,23
23,1,<,23
23, ,>,24
```

4.Процесс копирования ПЧ в конец строки

```
21,0,<,21
21,1,<,21
21, ,<,22
22,0,<,22
22,1,<,22
22, ,<,23
23,0,<,23
23,1,<,23
23, ,>,24
24,0, ,25
24,1, ,26
25, ,>,27
26, ,>,28
27,0,>,27
27,1,>,27
27, ,>,29
28,0,>,28
28,1,>,28
```

```
28, ,>,30
29,0,>,29
29,1,>,29
29, ,>,31
30,0,>,30
30,1,>,30
30, ,>,32
31,0,>,31
31,1,>,31
31, ,>,33
32,0,>,32
32,1,>,32
32, ,>,34
33, ,0,35
34, ,1,36
35,0,<,35
35,1,<,35
35, ,<,37
37,0,<,37
37,1,<,37
37, ,<,39
39,0,<,39
39,1,<,39
39, ,<,41
41,0,<,41
41,1,<,41
41, ,0,43
36,0,<,36
```

```
36,1,<,36
36, ,<,38
38,0,<,38
38,1,<,38
38, ,<,40
40,0,<,40
40,1,<,40
40, ,<,42
42,0,<,42
42,1,<,42
42, ,1,43
43, ,>,100
43,0,>,44
43,1,>,44
44,0, ,45
44,1, ,46
44, ,>,100
45, ,>,47
46, ,>,48
47,0,>,47
47,1,>,47
47, ,>,49
48,0,>,48
48,1,>,48
48, ,>,50
49,0,>,49
49,1,>,49
49, ,>,51
```

```
50,0,>,50
50,1,>,50
50, ,>,52
51,0,>,51
51,1,>,51
51, ,>,53
52,0,>,52
52,1,>,52
52, ,>,54
53,0,>,53
53,1,>,53
53, ,0,35
54,0,>,54
54,1,>,54
54, ,1,36
```

5. Движение по уже скопированным цифрам в конец строки и окончание алгоритма

```
100, ,>,101
100,0,>,100
100,1,>,100


101, ,>,102
101,0,>,101
101,1,>,101
102, ,=,103
102,0,>,102
102,1,>,102
103, ,#,103
```