Министерство образования Российской Федерации

Государственное образовательное учреждение

Московский авиационный институт

Факультет компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

КУРСОВАЯ РАБОТА

По дисциплине

«Фундаментальная информатика»

I семестр

Лабораторные работы

«Построение алгоритмических моделей на примере моделей Тьюринга и Маркова»

Студент: Соловьева Н. С.

Год приема: 2023

Группа: М8О-108Б-23

Проверил:

Преподаватель каф. 806 Севастьянов В. С.

Москва, 2023

Содержание:

Введение

Перед тем, как приступить непосредственно к практической части данной работы, необходимо ознакомиться с теоретической составляющей её темы.

Что такое алгоритм? Определено понятие алгоритма (или эффективной процедуры) как последовательности правил для получения определенного выходного сообщения из заданного входного сообщения за конечное число шагов. Важно, чтобы действия, указанные в алгоритме, были механическими, понятными и легко выполняемыми для всех исполнителей - людей и автоматов. Несмотря на практическую применимость данного определения, у него есть недостатки. Не всегда понятно, что означает "понимать и выполнять действия одинаково" или "понятные, легко выполнимые действия". Такие примеры как вычисление производной многочлена и завязывание шнурков, показывают, что интерпретация алгоритма может быть разной. В математике не все задачи могут быть решены с помощью алгоритма, и доказательство алгоритмической неразрешимости класса задач требует четкого формального определения алгоритма.

Формализация понятия алгоритма реализуется с помощью построения алгоритмических моделей. Можно выделить три основных типа универсальных алгоритмических моделей: рекурсивные Функции (понятие алгоритма связывается с вычислениями и числовыми Функциями), машины Тьюринга (алгоритм представляется как описание процесса работы некоторой машины, способной выполнять лишь небольшое число весьма простых операций), нормальные алгоритмы Маркова (алгоритмы описываются как преобразования слов в произвольных алфавитах).

Целью данной работы является демонстрация определения алгоритма с помощью построения алгоритмических моделей Тьюринга и Маркова.

В процессе мы рассмотрим несколько лабораторных работ, задачами которых являются:

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.

При выполнении работы использовались:

- Эмулятор машины Тьюринга в четвёрках (jstu4)
- Диаграммер для работы с диаграммами Тьюринга (VirtualTuringMachine)
- Эмулятор для нормальных алгоритмов Маркова (nam)

Вспомогательная литература:

- С. С. Гайсарян, В. Е. Зайцев «Курс информатики», Москва, Издательство Вузовская книга, 2013 г.

1. Машина Тьюринга

1.1 Теоретическая часть, связанная с МТ

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 ′∈ Q, a ∈A∪{λ}, v ∈ {l, r } ∪ A ∪ {λ} (программа), определяющее три функции: функцию выхода Fl: Q∗¯ A→¯A (ЃA =A∪{λ}),функцию переходов Ft :Q∗¯ A → Q, и функцию движения головки Fv : Q ∗¯A → {l, r, s} (символ s означает, что головка неподвижна).

Можно сказать, что каждая четвёрка состоит из:

1) начального состояния (то, в котором находится головка до выполнения следующей команды)
2) начальной буквы, расположенной там, где находится головка
3) команды, которую должна сделать машина Тьюринга. Это может быть:
   - переход на одну ячейку влево;
   - переход на одну ячейку вправо;
   - замена начальной буквы на другую;
4) состояния, в которое должна перейти головка.

Стоит учитывать, что если в четвёрке не описано какое-нибудь действие (команда или смена состояние), то это приведёт к зацикливанию программы и всё будет очень плохо.

Подводя итоги теоретической части, нужно сказать, что на вот этих «четвёрках» и построено определение алгоритма по Тьюрингу. С их помощью и задаются правила, по которым можно определить за конечное число шагов необходимое нам преобразование сообщения. Определение алгоритмов по Тьюрингу настолько фундаментально, что с его помощью можно описать всё, что только возможно представить в виде алгоритма.

1.2 Лабораторная работа

«Программирование машин Тьюринга»

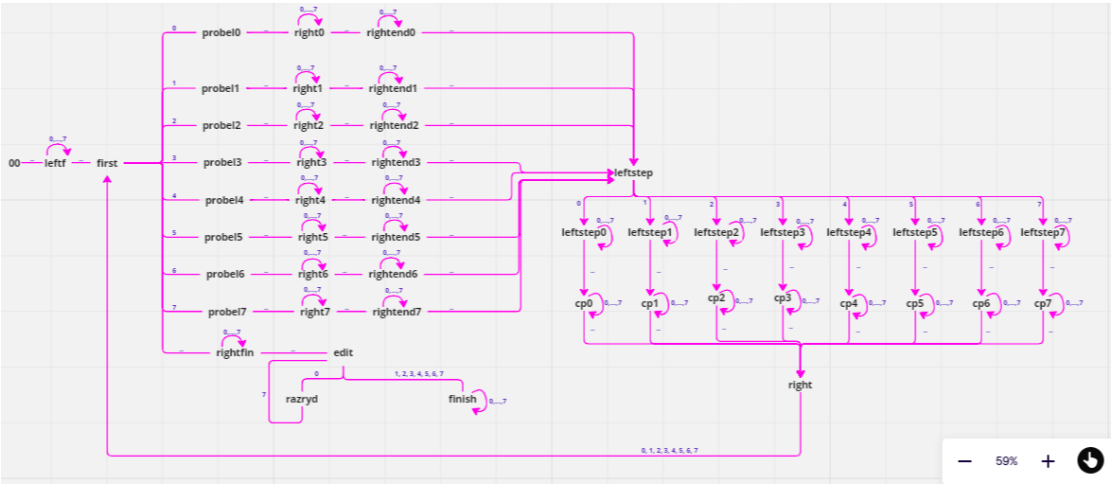
Цель работы: Составить программу машины Тьюринга в четвёрках, выполняющую заданное действие над словами, записанными на ленте. Отладку и тестирование проводить в среде интерактивного действующего макета jstu4. В начальном состоянии головка МТ находится на пустой ячейке непосредственно справа от записанных на ленте аргументов – слов входного сообщения. В конечном состоянии головка МТ должна находиться на пустой ячейке непосредственно справа от результата (последнего преобразованного или вновь сгенерированного слова результирующего сообщения). Вычисления в программе, как правило, должны быть нормированными (аргументы после работы программы сохраняются на ленте в неизменном виде и не остаётся промежуточных результатов.

Задание: Уменьшение на единицу целого неотрицательного числа в восьмеричной системе счисления.

Метод решения задачи: Посимвольно копируем вводное слово справа рядом через пробел. Далее изменяем записанное скопированное слово в зависимости от задания (меняем символы в соответствии с правилом вычитания единицы из числа в восьмеричной системе счисления).

Подробнее способы перехода состояний можно наблюдать на рис. 1

Рис.1



Команды и их функции**:**

- **00** - двигаем головку влево от стартовой позиции (пробел после вводного слова);
- **leftf** - двигаем головку влево до начала вводного слова (пробела);
- **checknull** – проверяем, является ли вводная строка 0;
- **nul** - пропускаем предстоящие нули вводного слова, двигая головку вправо;
- **otr** - если вводная строка 0, то выводим -1 в виде большого числа в 8 СС через команды otr1, ..., otr19;
- **first** - считываем первую после пробела цифру n и в зависимости от её значения выстраиваем следующий алгоритм:

  - **probeln** - заменяем цифру n на пробел;

  - **rightn** - двигаем головку вправо до конца вводного слова (пробела);

  - **rightendn** - двигаем головку вправо до конца создаваемого слова (пробела), в конце этого слова записываем цифру n;

  - **leftstep** - считываем последнюю цифру создаваемого слова n и в зависимости от её значения продолжаем алгоритм;

  - **leftstepn** - двигаем головку влево до начала создаваемого слова/конца вводного слова (пробела);

  - **cpn** - двигаем головку влево до пробела, которым мы командой probeln заменили цифру n, и меняем его на изначально стоящую здесь цифру n;

  - **right** - двигаем головку вправо и начинаем процесс копирования следующей цифры командой first;
- **rightfin** - если скопированы все цифры водного слова (команда first оказалась в конце вводного слова (пробеле)), двигаем головку вправо до конца созданного слова (пробела);
- **edit** - меняем цифру n на цифру n-1, если не происходит переход на следующий разряд;

- **razryd** - двигаем головку влево и меняем следующий разряд;
- **finish** - двигаем головку вправо до конца измененного скопированного слова (пробела) → конец

Сценарий выполнения работы:

Входные данные	Выходные данные	Описание тестируемого случая
767	767 766	Вычитание единицы без перехода на следующий разряд.
770	770 767	Вычитание единицы с переходом на следующий разряд.
700	700 677	Вычитание единицы с переходом на несколько следующих разрядов.
100	100 077	Вычитание единицы с изменением разрядности числа.
0	0 7777777777777777	Вычитание единицы с выводом -1 через большое положительное число.
00000001	00000001 0	Вычитание единицы с устранением предстоящих нулей.

Протокол:

00, ,<,leftf

leftf,1,<,leftf

leftf,2,<,leftf

leftf,0,<,checknul

leftf,3,<,leftf

leftf,4,<,leftf

leftf,5,<,leftf

leftf,6,<,leftf

leftf,7,<,leftf

leftf, ,>,first

checknul, ,>,nul

checknul,1,<,leftf

checknul,2,<,leftf

checknul,3,<,leftf

checknul,4,<,leftf

checknul,5,<,leftf

checknul,6,<,leftf

checknul,7,<,leftf

checknul,0,<,leftf

nul,0,>,nul

nul, ,>,otr

nul,1,1,first

nul,2,2,first

nul,3,3,first

nul,4,4,first

nul,5,5,first

nul,6,6,first

nul,7,7,first

otr, ,7,otr1

otr1,7,>,otr2

otr2, ,7,otr2

otr2,7,>,otr3

otr3, ,7,otr3

otr3,7,>,otr4

otr4, ,7,otr4

otr4,7,>,otr5

otr5, ,7,otr5

otr5,7,>,otr6

otr6, ,7,otr6

otr6,7,>,otr7

otr7, ,7,otr7

otr7,7,>,otr8

otr8, ,7,otr8

otr8,7,>,otr9

otr9, ,7,otr9

otr9,7,>,otr10

otr10, ,7,otr10

otr10,7,>,otr11

otr11, ,7,otr11

otr11,7,>,otr12

otr12, ,7,otr12

otr12,7,>,otr13

otr13, ,7,otr13

otr13,7,>,otr14

otr14, ,7,otr14

otr14,7,>,otr15

otr15, ,7,otr15

otr15,7,>,otr16

otr16, ,7,otr16

otr16,7,>,otr17

otr17, ,7,otr17

otr17,7,>,otr18

otr18, ,7,otr18

otr18,7,>,otr19

otr19, ,7,finish

right1,0,>,right1
right1,2,>,right1
right1,1,>,right1
right1,3,>,right1
right1,4,>,right1
right1,5,>,right1
right1,6,>,right1
right1,7,>,right1
right1, ,>,rightend1

right0,0,>,right0
right0,2,>,right0
right0,1,>,right0
right0,3,>,right0
right0,4,>,right0
right0,5,>,right0
right0,6,>,right0
right0,7,>,right0
right0, ,>,rightend0

right2,0,>,right2
right2,2,>,right2
right2,1,>,right2
right2,3,>,right2
right2,4,>,right2
right2,5,>,right2
right2,6,>,right2
right2,7,>,right2
right2, ,>,rightend2

right3,0,>,right3

right3,2,>,right3

right3,1,>,right3

right3,3,>,right3

right3,4,>,right3

right3,5,>,right3

right3,6,>,right3

right3,7,>,right3

right3, ,>,rightend3

right4,0,>,right4

right4,2,>,right4

right4,1,>,right4

right4,3,>,right4

right4,4,>,right4

right4,5,>,right4

right4,6,>,right4

right4,7,>,right4

right4, ,>,rightend4

right5,0,>,right5

right5,2,>,right5

right5,1,>,right5

right5,3,>,right5

right5,4,>,right5

right5,5,>,right5

right5,6,>,right5

right5,7,>,right5

right5, ,>,rightend5

right6,0,>,right6

right6,2,>,right6

right6,1,>,right6

right6,3,>,right6

right6,4,>,right6

right6,5,>,right6

right6,6,>,right6

right6,7,>,right6

right6, ,>,rightend6

right7,0,>,right7

right7,2,>,right7

right7,1,>,right7

right7,3,>,right7

right7,4,>,right7

right7,5,>,right7

right7,6,>,right7

right7,7,>,right7

right7, ,>,rightend7

rightend1,0,>,rightend1

rightend1,2,>,rightend1

rightend1,1,>,rightend1

rightend1,3,>,rightend1

rightend1,4,>,rightend1

rightend1,5,>,rightend1

rightend1,6,>,rightend1

rightend1,7,>,rightend1

rightend1, ,1,leftstep

rightend0,0,>,rightend0

rightend0,2,>,rightend0

rightend0,1,>,rightend0

rightend0,3,>,rightend0

rightend0,4,>,rightend0

rightend0,5,>,rightend0

rightend0,6,>,rightend0

rightend0,7,>,rightend0

rightend0, ,0,leftstep

rightend2,0,>,rightend2

rightend2,2,>,rightend2

rightend2,1,>,rightend2

rightend2,3,>,rightend2

rightend2,4,>,rightend2

rightend2,5,>,rightend2

rightend2,6,>,rightend2

rightend2,7,>,rightend2

rightend2, ,2,leftstep

rightend3,0,>,rightend3

rightend3,2,>,rightend3

rightend3,1,>,rightend3

rightend3,3,>,rightend3

rightend3,4,>,rightend3

rightend3,5,>,rightend3

rightend3,6,>,rightend3

rightend3,7,>,rightend3

rightend3, ,3,leftstep

rightend4,0,>,rightend4

rightend4,2,>,rightend4

rightend4,1,>,rightend4

rightend4,3,>,rightend4

rightend4,4,>,rightend4

rightend4,5,>,rightend4

rightend4,6,>,rightend4

rightend4,7,>,rightend4

rightend4, ,4,leftstep

rightend5,0,>,rightend5

rightend5,2,>,rightend5

rightend5,1,>,rightend5

rightend5,3,>,rightend5

rightend5,4,>,rightend5

rightend5,5,>,rightend5

rightend5,6,>,rightend5

rightend5,7,>,rightend5

rightend5, ,5,leftstep

rightend6,0,>,rightend6

rightend6,2,>,rightend6

rightend6,1,>,rightend6

rightend6,3,>,rightend6

rightend6,4,>,rightend6

rightend6,5,>,rightend6

rightend6,6,>,rightend6

rightend6,7,>,rightend6

rightend6, ,6,leftstep

rightend7,0,>,rightend7

rightend7,2,>,rightend7

rightend7,1,>,rightend7

rightend7,3,>,rightend7

rightend7,4,>,rightend7

rightend7,5,>,rightend7

rightend7,6,>,rightend7

rightend7,7,>,rightend7

rightend7, ,7,leftstep

first,1, ,probel1

first,0, ,probel0

first,2, ,probel2

first,3, ,probel3

first,4, ,probel4

first,5, ,probel5

first,6, ,probel6

first,7, ,probel7

first, ,>,rightfin

probel1, ,>,right1

probel0, ,>,right0

probel2, ,>,right2

probel3, ,>,right3

probel4, ,>,right4

probel5, ,>,right5

probel6, ,>,right6

probel7, ,>,right7

leftstep,1,<,leftstep1

leftstep,2,<,leftstep2

leftstep,3,<,leftstep3

leftstep,4,<,leftstep4

leftstep,5,<,leftstep5

leftstep,6,<,leftstep6

leftstep,7,<,leftstep7

leftstep,0,<,leftstep0

leftstep1,1,<,leftstep1

leftstep1,2,<,leftstep1

leftstep1,3,<,leftstep1

leftstep1,4,<,leftstep1

leftstep1,5,<,leftstep1

leftstep1,6,<,leftstep1

leftstep1,7,<,leftstep1

leftstep1,0,<,leftstep1

leftstep1, ,<,cp1

leftstep2,1,<,leftstep2

leftstep2,2,<,leftstep2

leftstep2,3,<,leftstep2

leftstep2,4,<,leftstep2

leftstep2,5,<,leftstep2

leftstep2,6,<,leftstep2

leftstep2,7,<,leftstep2

leftstep2,0,<,leftstep2

leftstep2, ,<,cp2

leftstep3,1,<,leftstep3

leftstep3,2,<,leftstep3

leftstep3,3,<,leftstep3

leftstep3,4,<,leftstep3

leftstep3,5,<,leftstep3

leftstep3,6,<,leftstep3

leftstep3,7,<,leftstep3

leftstep3,0,<,leftstep3

leftstep3, ,<,cp3

leftstep4,1,<,leftstep4

leftstep4,2,<,leftstep4

leftstep4,3,<,leftstep4

leftstep4,4,<,leftstep4

leftstep4,5,<,leftstep4

leftstep4,6,<,leftstep4

leftstep4,7,<,leftstep4

leftstep4,0,<,leftstep4

leftstep4, ,<,cp4

leftstep5,1,<,leftstep5

leftstep5,2,<,leftstep5

leftstep5,3,<,leftstep5

leftstep5,4,<,leftstep5

leftstep5,5,<,leftstep5

leftstep5,6,<,leftstep5

leftstep5,7,<,leftstep5

leftstep5,0,<,leftstep5

leftstep5, ,<,cp5

leftstep6,1,<,leftstep6

leftstep6,2,<,leftstep6

leftstep6,3,<,leftstep6

leftstep6,4,<,leftstep6

leftstep6,5,<,leftstep6

leftstep6,6,<,leftstep6

leftstep6,7,<,leftstep6

leftstep6,0,<,leftstep6

leftstep6, ,<,cp6

leftstep7,1,<,leftstep7

leftstep7,2,<,leftstep7

leftstep7,3,<,leftstep7

leftstep7,4,<,leftstep7

leftstep7,5,<,leftstep7

leftstep7,6,<,leftstep7

leftstep7,7,<,leftstep7

leftstep7,0,<,leftstep7

leftstep7, ,<,cp7

leftstep0,1,<,leftstep0

leftstep0,2,<,leftstep0

leftstep0,3,<,leftstep0

leftstep0,4,<,leftstep0

leftstep0,5,<,leftstep0

leftstep0,6,<,leftstep0

leftstep0,7,<,leftstep0

leftstep0,0,<,leftstep0

leftstep0, ,<,cp0

cp1,1,<,cp1

cp1,2,<,cp1

cp1,3,<,cp1

cp1,4,<,cp1

cp1,5,<,cp1

cp1,6,<,cp1

cp1,7,<,cp1

cp1,0,<,cp1

cp1, ,1,right

cp2,1,<,cp2

cp2,2,<,cp2

cp2,3,<,cp2

cp2,4,<,cp2

cp2,5,<,cp2

cp2,6,<,cp2

cp2,7,<,cp2

cp2,0,<,cp2

cp2, ,2,right

cp3,1,<,cp3

cp3,2,<,cp3

cp3,3,<,cp3

cp3,4,<,cp3

cp3,5,<,cp3

cp3,6,<,cp3

cp3,7,<,cp3

cp3,0,<,cp3

cp3, ,3,right

cp4,1,<,cp4

cp4,2,<,cp4

cp4,3,<,cp4

cp4,4,<,cp4

cp4,5,<,cp4

cp4,6,<,cp4

cp4,7,<,cp4

cp4,0,<,cp4

cp4, ,4,right

cp5,1,<,cp5

cp5,2,<,cp5

cp5,3,<,cp5

cp5,4,<,cp5

cp5,5,<,cp5

cp5,6,<,cp5

cp5,7,<,cp5

cp5,0,<,cp5

cp5, ,5,right

cp6,1,<,cp6

cp6,2,<,cp6

cp6,3,<,cp6

cp6,4,<,cp6

cp6,5,<,cp6

cp6,6,<,cp6

cp6,7,<,cp6

cp6,0,<,cp6

cp6, ,6,right

cp7,1,<,cp7

cp7,2,<,cp7

cp7,3,<,cp7

cp7,4,<,cp7

cp7,5,<,cp7

cp7,6,<,cp7

cp7,7,<,cp7

cp7,0,<,cp7

cp7, ,7,right

cp0,1,<,cp0

cp0,2,<,cp0

cp0,3,<,cp0

cp0,4,<,cp0

cp0,5,<,cp0

cp0,6,<,cp0

cp0,7,<,cp0

cp0,0,<,cp0

cp0, ,0,right

right,1,>,first

right,2,>,first

right,3,>,first

right,4,>,first

right,5,>,first

right,6,>,first

right,7,>,first

right,0,>,first

rightfin,1,>,rightfin

rightfin,2,>,rightfin

rightfin,3,>,rightfin

rightfin,4,>,rightfin

rightfin,5,>,rightfin

rightfin,6,>,rightfin

rightfin,7,>,rightfin

rightfin,0,>,rightfin

rightfin, ,<,edit

edit,0,7,razryd

edit,7,6,finish

edit,6,5,finish

edit,5,4,finish

edit,4,3,finish

edit,3,2,finish

edit,2,1,finish

edit,1,0,finish

finish,1,>,finish

finish,2,>,finish

finish,3,>,finish

finish,4,>,finish

finish,5,>,finish

finish,6,>,finish

finish,7,>,finish

finish,0,>,finish

finish, , ,finish

razryd,7,<,edit

1.3 Выводы

Подводя итог данной демонстрации работы алгоритмической модели Тьюринга можно сделать следующие выводы:

1. Машины Тьюринга представляют собой мощную абстрактную модель вычислений. Они позволяют описывать и эмулировать широкий спектр алгоритмов и вычислительных процессов.
2. Машины Тьюринга демонстрируют универсальность - они способны эмулировать работу других систем и являются эквивалентными по вычислительной мощности с другими моделями, такими как рекурсивные функции или нормальные алгоритмы Маркова.
3. Машины Тьюринга предлагают простую архитектуру, основанную на ленте, на которой выполняются операции чтения и записи. Это делает их понятными и удобными для моделирования и анализа различных алгоритмических задач.
4. Машины Тьюринга имеют свои ограничения, такие как ограниченность памяти и времени работы. Однако, несмотря на эти ограничения, они остаются важным инструментом для теоретического исследования вычислительной сложности и алгоритмической теории.
5. Машины Тьюринга являются основой для понимания понятий вычислимости и алгоритма. Они помогли разработать формальные критерии для определения вычислительной сложности и являются теоретическим фундаментом для развития современных компьютерных наук.

2. Диаграмма Тьюринга

2.1 Теоретическая часть, связанная с ДТ

Стоит обратить внимание на то, что стандартные машины Тьюринга имеют значительные недостатки:

- Для каждого шага и малейшего сдвига головки приходиться прописывать огромное количество состояний, что значительно замедляет работу;
- Машина Тьюринга визуально трудна для восприятия, так как представляет собой огромное количество текста и состояний;
- Большинство задач выполняется через копирование вводный данных, написание состояний для которого занимает большую часть от времени, потраченного на всю работу.

Для решения этих проблем были созданы диаграммы Тьюринга.

Диаграммы Тьюринга представляют одну МТ через другую, более простую МТ, визуально-топологическим способом. Этот способ так же строг и полон, как и "обычные" МТ. Например, машина, копирующая слово на ленте, может быть представлена через МТ, ищущую начало и конец слова на ленте, выполняющую копирование букв слова и т.д. Более простые МТ, в свою очередь, могут быть представлены еще более простыми МТ, и так далее.

Такой процесс представления МТ через более простые МТ должен обязательно прекратиться, сводя описание каждой МТ к элементарным действиям, определенным при описании МТ. Результатом будет представление МТ через элементарные МТ, которые выполняют только одно элементарное действие и останавливаются.

Элементарные МТ решают вышеописанные проблемы. Топографический формат записи и возможность разбивать программу на части устраняют второй недостаток "обычных" машин Тьюринга.

2.2 Лабораторная работа

«Конструирование диаграмм Тьюринга»

Цель работы: Разработать диаграмму Тьюринга решения задачи в среде интерпретатора jdt или VisualTuring 2.0 с использованием стандартных машин (r, l, R, L, K, a) и вспомогательных машин, определяемых поставленной задачей. Вариант даётся преподавателем согласно выбранному студентом уровню сложности. Алфавит диаграммы определяется заданием. В начальном состоянии голова МТ, определяемой диаграммой, находится на пустой ячейке непосредственно справа от записанных на ленте аргументов. В конечном состоянии головка МТ должна находиться на пустой ячейке непосредственно справа от результата (последнего преобразованного или вновь сформированного слова). Определяемый заданием вычисления должны быть нормированными (аргументов после работы программы сохраняются на ленте в неизменном виде и не остаётся промежуточных результатов).

Задание: Зеркальное отражение двух десятичных слов относительно промежутка между ними.

Метод решения задачи: Посимвольно копируем второе введенное слово с конца до разделительного пробела путём замены символов в этом слове на пробелы и их дальнейшего возвращения. Таким образом дописываем в конце строки отзеркаленное второе слово. В результате предыдущих действий головка оказывается на разделительном пробеле. Далее повторяем такой же алгоритм, меняя дистанцию отзеркаливания, и посимвольно копируем первое введенное слово с конца до первостоящего пробела. Затем четырежды выполняем R и оказываемся на конечной позиции - пробеле после отзеркаленной строки --> Конец. Диаграмма реализует нормированный вывод без использования сторонних знаков (не входящих в алфавит задания).

Вспомогательные машины и их функции**:**

- ***Mirror last symbol in second word*** - Копирование последнего символа второго введенного слова через пробел.
- ***Mirror other symbols in second word*** - Посимвольное отзеркаливание остальных символов второго введенного слова с конца до разделительного пробела между введенными словами путём замены цифр на пробел, дописывания их в конце строки и возвращения замененного символа во вводном слове.
- ***Mirror last symbol in first word*** - аналогично машине 'Mirror last symbol in first word' для копирования последнего символа первого введенного слова в конце строки.
- ***Mirror other symbols in first word*** - аналогично машине 'Mirror other symbols in second word' для посимвольного отзеркаливания первого введенного слова и записи его в конце строки.

Сценарий выполнения работы:

Входные данные	Выходные данные	Описание тестируемого случая
123 456	123 456 654 321	Зеркального вывода десятичных слов разных знаков одной длины
777 666	777 666 666 777	Зеркальный вывод десятичных слов состоящих из разных единственных цифр одной длины
010 010	010 010 010 010	Зеркальный вывод десятичных слов-палиндромов
5 10000	5 10000 00001 5	Зеркальный вывод десятичных слов разных знаков разной длины

Протокол:

Общая диаграмма Тьюринга (рис.2);

Вспомогательная машина: “Mirror last symbol in second word” (рис. 3);

Вспомогательная машина: "Mirror other symbols in second word" (рис. 4);

Вспомогательная машина: “Mirror last symbol in first word” (рис. 5);

Вспомогательная машина: “Mirror other symbols in first word” (рис. 6).

Рис. 2

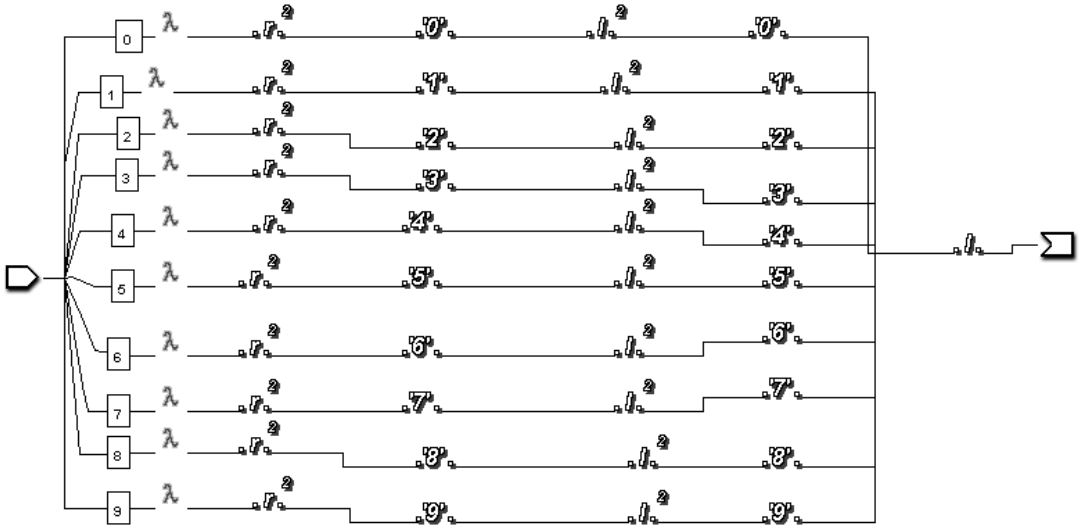


Рис. 3

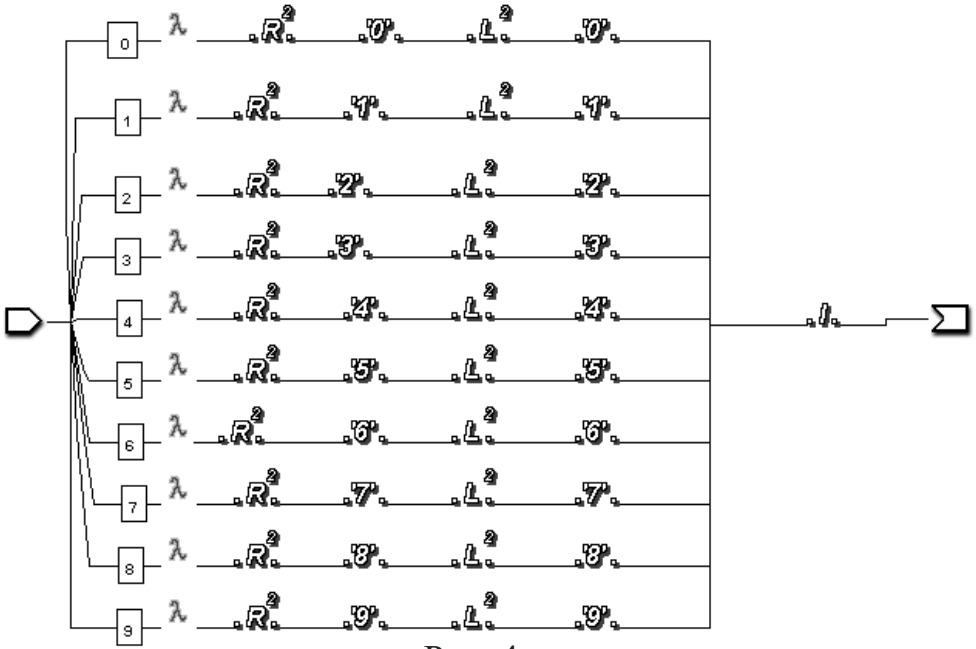


Рис. 4

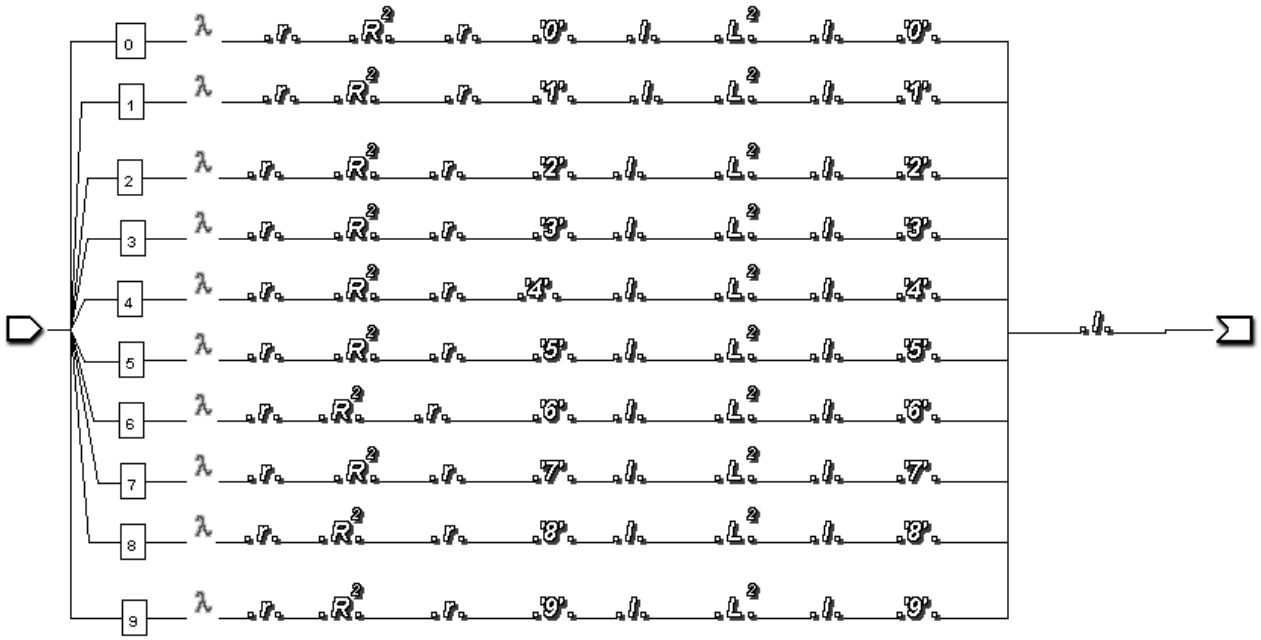


Рис. 5

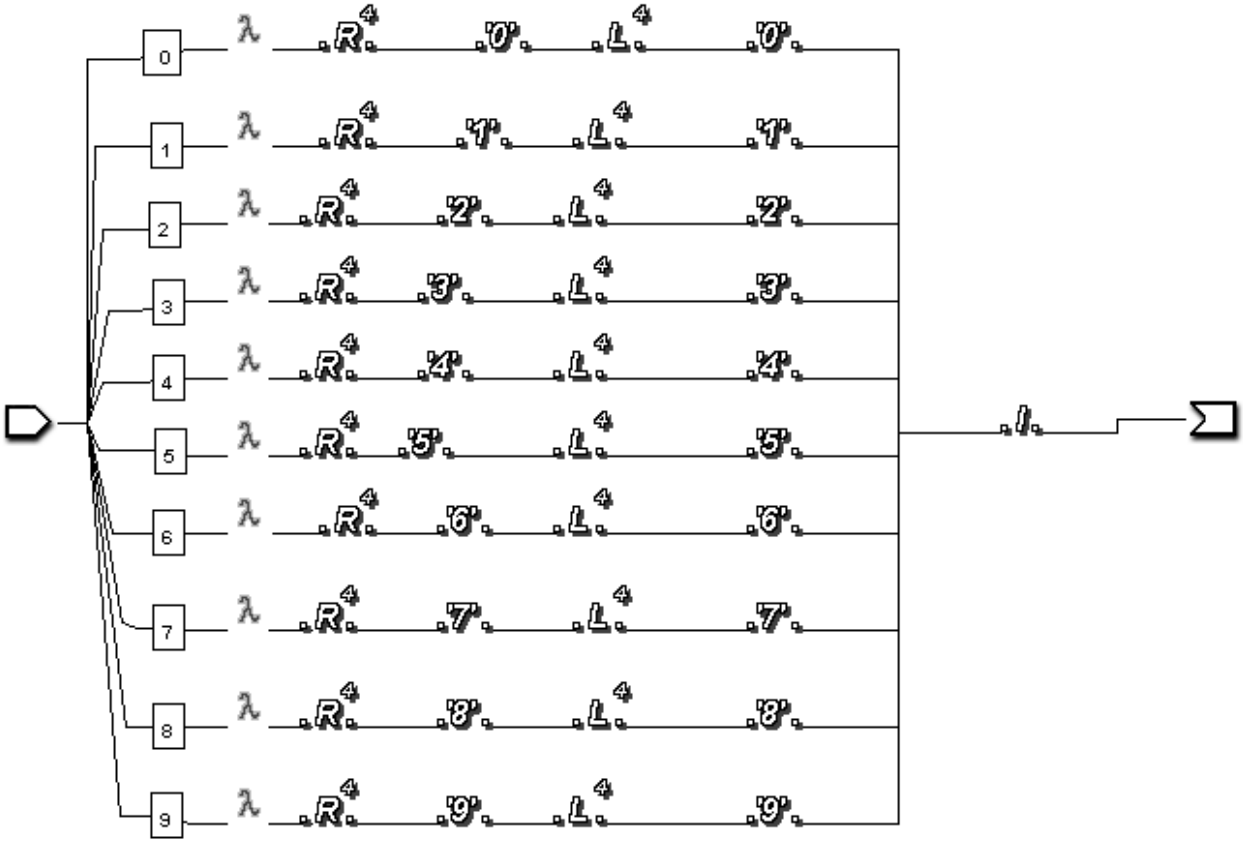


Рис. 6

2.3 Выводы

Подводя итог данной демонстрации алгоритмической модели Тьюринга в виде диаграммы Тьюринга, можно сделать следующие выводы:

1. Диаграммы Тьюринга представляют собой мощный инструмент для визуализации и анализа работы машин Тьюринга. Они позволяют упростить и улучшить понимание сложных процессов, которые выполняются на машинах Тьюринга.
2. Диаграммы Тьюринга предоставляют возможность лучшего представления работы МТ через более простые элементарные действия. Это позволяет уменьшить размер и сложность программы, улучшить ее читаемость и обеспечить более эффективное исполнение.
3. Использование диаграмм Тьюринга позволяет лучше структурировать и организовать работу с машинами Тьюринга. Они помогают разделить процессы на более небольшие этапы и логически связать их друг с другом, что упрощает отладку и модификацию программы.
4. Один из важных аспектов диаграмм Тьюринга – это их наглядность. Они позволяют визуально представить состояния, переходы и действия, выполняемые машиной Тьюринга. Это помогает не только разработчикам понять работу программы, но и быть более доступными для других заинтересованных лиц.
5. Диаграммы Тьюринга могут быть использованы для обучения и образовательных целей. Они позволяют студентам лучше понять основные принципы и концепции, связанные с машинами Тьюринга, и получить практические навыки в их разработке и анализе.

В целом, диаграммы Тьюринга представляют собой важный инструмент в исследованиях и применении машин Тьюринга. Они облегчают анализ и визуализацию работы МТ, улучшают структуру программ и обеспечивают лучшее понимание их функционирования.

3. Нормальные алгоритмы Маркова (НАМ)

3.1 Теоретическая часть

Нормальный алгоритм Маркова (НАМ) представляет собой набор правил-продукций, состоящих из пар слов (цепочек знаков), объединенных символами "->" или "→". Продукции задают правила замены части входного слова на их правую часть. Каждая продукция может иметь пустую левую или правую часть. При применении продукции, вхождение левой части заменяется на правую, и процесс начинается заново. Если ни одна из продукций не применима, проверяется следующая продукция в порядке их описания. Выполнение алгоритма завершается, если все продукции неприменимы или применяется завершающая продукция с символом "->". Терминальных продукций может быть несколько. Порядок обработки правил определяется первым применяемым правилом и выбором самого левого вхождения, если правило применимо в нескольких местах. НАМ применим ко всем входным словам, если левые части продукций непустые и правые части короче левых.

Эти правила и признаки были введены из-за специфики нормальных алгоритмов Маркова, которые не имеют головки, курсора или какого-либо другого механизма для указания текущей позиции входного слова, как в машинах Тьюринга или диаграммах Тьюринга. Это отличие приводит к существенным различиям в реализации алгоритмов Маркова и выводе результатов (в том числе, формат вывода может быть ненормированным, так что не требуется копирование).

Таким образом, указанные признаки и правила значительно влияют на особенности работы с нормальными алгоритмами Маркова и их специфику.

3.2 Лабораторная работа

«Программирование в алгоритмической модели Маркова»

Цель работы: Написать алгоритм Маркова, выполняющий задание, установленное вариантом (ненормированный вывод).

Задание: Составить алгоритм, восстанавливающий целое число в шестнадцатеричной позиционной системе счисления по его дополнительному коду.

Метод решения задачи:
Вводная строка - число в 16-ной системе счисления.

- Перед первым символом ставим переменную, обозначающую начало строки.
- Если первый символ ∈ {0, 1, 2, 3, 4, 5, 6, 7}, то первый символ дополнительного кода будет 0. Следовательно, число положительное. В этом случае оставляем вводную строку без изменений, удаляя введенную переменную.
- Если первый символ ∈ {8, 9, A, B, C, D, E, F}, то первый символ дополнительного кода будет 1. Следовательно, число отрицательное. В этом случае выполняем следующий алгоритм:
    - Переводим 16-ную число в двоичную систему счисления, разделяя каждые четыре знака промежуточной переменной * (в результате преобразований символ \ оказывается в конце строки).
    - Вычитаем единицу из последней четверки вида *XXXX*\ полученного двоичного числа и меняем переменную \ на |.
    - Перемещаем переменную | в начало строки.
    - Заменяем каждую четверку двоичных символов вида *XXXX* на соответствующее ему инвертированное число в 16-ной системе счисления.
    - Меняем стоящую вначале строки переменную | на -.
    - Удаляем все разделительные переменные *.

Сценарий выполнения работы:

Входные данные	Выходные данные	Описание тестируемого случая
A4	-5C	Восстановление отрицательного числа в 16-ной системе счисления по его доп. коду
71	71	Восстановление положительного числа в 16-ной системе счисления по его доп. коду
0	0	Восстановление нуля в 16-ной системе счисления по его доп. коду
00000	0	Восстановление нуля с удалением лишних нулей в 16-ной системе счисления по его доп. коду
FFFFF	-1	Восстановление отрицательного числа с удалением ненужных нулей в 16-ной системе счисления по его доп. коду
E0	-20	Восстановление отрицательного числа с переходом между разрядами при вычитании единицы в 16-ной системе счисления по его доп. коду
0000111	111	Восстановление положительного числа с удалением ненужных нулей в 16-ной системе счисления по его доп. коду

Протокол:

*\0->*0000*\

*\1->*0001*\

*\2->*0010*\

*\3->*0011*\

*\4->*0100*\

*\5->*0101*\

*\6->*0110*\

*\7->*0111*\

\0->\

\1->.1

\2->.2

\3->.3

\4->.4

\5->.5

\6->.6

\7->.7

\8->*1000*\

\9->*1001*\

\A->*1010*\

\B->*1011*\

\C->*1100*\

\D->*1101*\

\E->*1110*\

\F->*1111*\

0000\->*1111*

0001\->*0000*|

0010\->*0001*|

0011\->*0010*|

0100\->*0011*|

0101\->*0100*|

0110\->*0101*|

0111\->*0110*|

1000\->*0111*|

1001\->*1000*|

1010\->*1001*|

1011\->*1010*|

1100\->*1011*|

1101\->*1100*|

1110\->*1101*|

1111\->*1110*|

|->|

0|->|0

1|->|1

0000->*F*

0001->*E*

0010->*D*

0011->*C*

0100->*B*

0101->*A*

0110->*9*

0111->*8*

1000->*7*

1001->*6*

1010->*5*

1011->*4*

1100->*3*

1101->*2*

1110->*1*

1111->*0*

|->-

*->

-0->-

-->.-

->\

3.3 Выводы

Подводя итог данной демонстрации алгоритмической модели Маркова, можно сделать следующие выводы:

1. Нормальные алгоритмы Маркова представляют собой эффективный математический инструмент, используемый для моделирования и анализа последовательностей символов.
2. НАМ применимы к широкому спектру задач, таким как обработка текстов, генерация кода, автоматическая классификация и т.д.
3. Основные преимущества НАМ включают простоту реализации, компактность представления и возможность сохранения состояния между применениями продукций.
4. Для успешного применения НАМ необходимо учитывать правила и признаки применимости, такие как непустые левые части продукций и короткие правые части.
5. Реализация алгоритма Маркова может отличаться от других моделей, таких как машины Тьюринга, и требовать определенных модификаций при обработке входной информации и формате вывода.
6. Использование НАМ в практических приложениях может значительно упростить задачи моделирования и автоматизации, предоставляя эффективные инструменты для обработки и анализа символьных последовательностей.

НАМ являются активной областью исследований, и дальнейшие исследования в этой области могут принести новые методики и подходы, расширяющие возможности и применимость нормальных алгоритмов Маркова.

Заключение

В данной курсовой работе были рассмотрены алгоритмы Тьюринга и Маркова. Было показано, что определение алгоритма может отличаться для каждого из них:

- В случае с машиной Тьюринга алгоритмом является набор состояний и действий, которые выполняются головкой для решения задачи.
- При построении диаграмм Тьюринга используется последовательность элементарных машин и программ, объединенных определенным образом.
- В нормальных алгоритмах Маркова результат достигается последовательным выполнением правил, которые преобразуют входную строку.

Верность этих рассуждений удалось продемонстрировать ходе выполнения лабораторных работ, выполнив все поставленные задачи успешно.

Работа выполнена в рамках требуемого объема и удовлетворяет всем требованиям.