МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационных систем и технологий

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_5_» _____ 2020 г.

ТЕОРИЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ И СТРУКТУР

Методические указания по выполнению лабораторных работ
для бакалавров направления подготовки 02.03.03
«Математическое обеспечение и администрирование информационных систем»

Курск 2020

УДК 621.(076.1)

Составитель В.П. Добрица, Е.А. Кулешова

Рецензент

Кандидат технических наук, доцент кафедры «Информационных систем и технологий» Ю.А. Халин

Теория вычислительных процессов и структур: методические указания по выполнению лабораторных работ для бакалавров направления подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем» / Юго-Зап. гос. ун-т; сост.: В.П. Добрица, Е.А. Кулешова. - Курск, 2020. - 39 с.: табл. 6 - Библиогр.: с. 39.

Методические указания соответствуют требованиям стандарта, утвержденного учебно-методическим объединением по специальности.

Предназначены для бакалавров направления подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем».

Текст печатается в авторской редакции

Подписано в печать 5.03.20. Формат 60x84 1/16.
Усл.печ. л.3. Уч.-изд. л. 2,7. Тираж 100 экз. Заказ. 59
Бесплатно.
Юго-Западный государственный университет.
305040, г.Курск, ул. 50 лет Октября, 94.

ПРЕДИСЛОВИЕ

В данных методических рекомендациях изложены материалы для курса «Теория вычислительных процессов и систем».

Целью дисциплины является приобретение обучаемым фундаментальных знаний в области теории вычислительных процессов и структур и выработка практических навыков применения этих знаний. Изучение основных положений теории вычислительных процессов и структур, их применения при создании трансляторов с различных языков программирования и разработке прикладных информационных систем, а также концепций автоматного программирования.

Рассмотрены следующие темы: Способы представления автомата. Синтез комбинационных автоматов. Детерминированные автоматы. Эквивалентные состояния и автоматы

По каждой теме представлены:

1) краткие теоретические положения;

2) перечень вопросов, выносимых на лабораторное задание;

3) примеры задач, выносимых на лабораторное занятие;

4) задачи, выносимые на самостоятельную работу студентов.

Данные методические рекомендации предназначены для проведения лабораторных занятий по дисциплине «Теория вычислительных процессов и систем» для студентов направления подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем».

Лабораторная работа № 1: Способы представления автомата. Синтез комбинационных автоматов

Цель работы: научиться представлять состояния конечных автоматов с помощью таблицы и диаграммы, составлять схемы автоматов по логическим функциям. Булевых функций одного аргумента четыре, а двух аргументов — шестнадцать и т. д.

Теория автоматов – раздел дискретной математики, изучающий математические модели реальных (технических, биологических, экономических) или возможных устройств, перерабатывающих дискретную информацию дискретными временными тактами.

Автомат – это устройство, предназначенное для выполнения целенаправленных действий без участия человека, рассматриваемый либо как реализующий ту или иную формальную грамматику (абстрактный автомат), либо как множество элементов и схема их соединения (структурный автомат).

Автоматом Мили называется система A = (U, Q, V, δ, λ), где множество U={ u1 ,...,un } – входной алфавит, его элементы – входные сигналы, Q= { q1,...qm} –множество внутренних состояний, множество V ={v1,...,vk } – выходной алфавит, его элементы – выходные сигналы, δ:U×Q→Q – функция переходов, λ: U×Q →V - функция выхода.

С конечным автоматом ассоциируется воображаемое устройство, которое может находиться в состоянии из множества Q, воспринимать сигналы из множества U и выдавать сигналы из множества V.

Схема называется комбинационной, если каждую из ее выходов можно представить как булеву функцию входных переменных, типа И-НЕ, И, ИЛИ, ИЛИ-НЕ и т.д.

Графическое изображение комбинационной схемы, при котором показаны связи между различными элементами (вентелями), а сами элементы представлены условными обозначениями, называется функциональной схемой.

В ходе разработки комбинационных схем приходится решать задачи анализа и синтеза. Задача анализа комбинационной схемы состоит в определении статических и динамических свойств комбинационной схемы. Задача синтеза комбинационной схемы заключается в построении из заданного набора логических элементов комбинационной схемы, реализующей заданную систему булевых функций.

Примеры выполнения заданий:

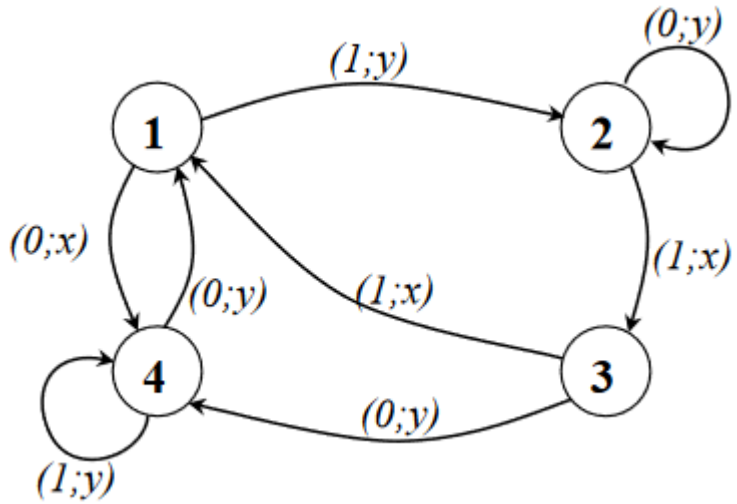
1. Для автоматов, заданных таблицами, построить диаграммы состояний:

U\Q	*1*	*2*	*3*	*4*
0	*4; x*	*2; y*	*4; y*	*1; y*
1	*2; y*	*3; x*	*1; x*	*4; y*

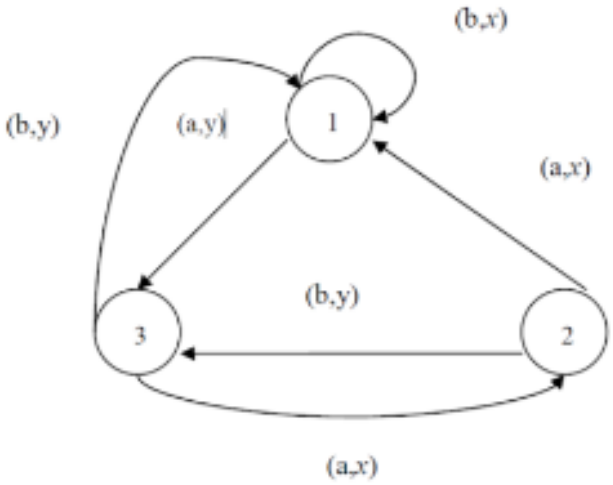
Решение:

- Автомат имеет четыре состояния, его множество состояний Q = {1, 2, 3, 4}, следовательно, граф автомата будет с четырьмя вершинами.
- Входной алфавит U = {0, 1}, а выходной алфавит V = {x, y}
- В каждой ячейке два объекта, первый соответствует направлению дуги из текущей вершины графа
- Над дугой записываем пару значений: значение входного алфавита и выходной символ из ячейки

Получим граф:



2. Задать табличным способом автомат, заданный диаграммой состояний:



Решение:

- Автомат имеет три состояния, его множество состояний Q = {1, 2, 3}, следовательно, в таблице четыре колонки.

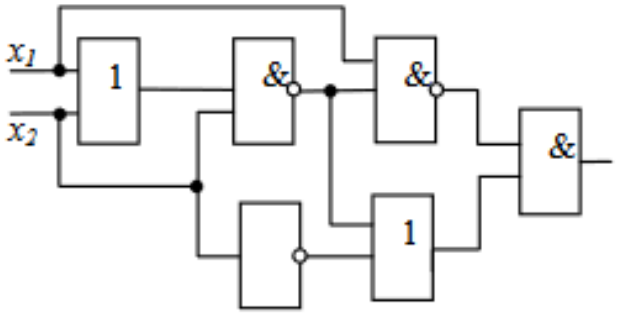
- Над дугами пары объектов, первый – элемент входного алфавита, второй –выходного, значит входной алфавит U = {a, b}, а выходной алфавит V = {x, y}, в первой колонке две строки a и b.

- В каждой ячейке два объекта, первый соответствует направлению дуги из текущей вершины графа, а второй – второму числу над соответствующей дугой.

Получим таблицу:

U\Q	*1*	*2*	*3*
a	*1; x*	*1; x*	*2; x*
b	*3; y*	*3; y*	*1; y*

3. Составить логическое выражение по схеме, упростить его и составить новую схему:



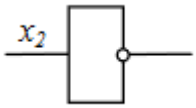
Решение:

- Следуем по схеме по верхней ветке: $w_1 = \overline{\overline{(x_1 \vee x_2)x_2}x_1} = \overline{\overline{(x_1 \vee x_2)} \vee \overline{x_1})x_1} = \overline{\overline{x_1 x_2} \vee \overline{x_1})x_1} = \overline{\overline{x_1}x_1} = 1$

- Теперь по нижней: $w_2 = \overline{(x_1 \vee x_2)x_2} \vee \overline{x_2} = \overline{(x_1 \vee x_2 \vee \overline{x_2})} \vee \overline{x_2} = \overline{x_1 x_2} \vee \overline{x_2} = \overline{x_2}$

- Объединим обе ветки: $w_1 w_2 = 1 * \overline{x_2}$

Получим схему:

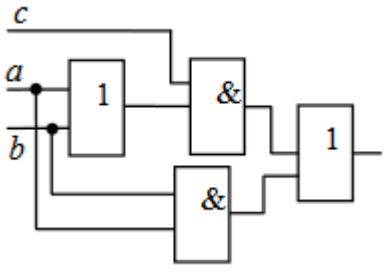


4. Для логической функции F(a, b, c), заданной в виде таблицы, записать аналитическое выражение и построить комбинационную схему:

a	b	c	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Решение:

- Составим СДНФ функции: $\bar{a}bc \vee a\bar{b}c \vee abc \vee abc$
- Минимизируем функцию: $\bar{a}bc \vee a\bar{b}c \vee abc \vee abc \equiv \bar{a}bc \vee abc \vee a\bar{b}c \vee abc \vee abc \vee abc \equiv bc \vee ac \vee ab \equiv c(a \vee b) \vee ab$
- Построим схему по минимизированному выражению:



Вопросы для самоконтроля:

1) Что такое автомат?
2) Что такое автомат Мили?
3) Что такое комбинационная схема автомата?
4) В чем суть анализа комбинационной схемы?
5) В чем суть синтеза комбинационной схемы?

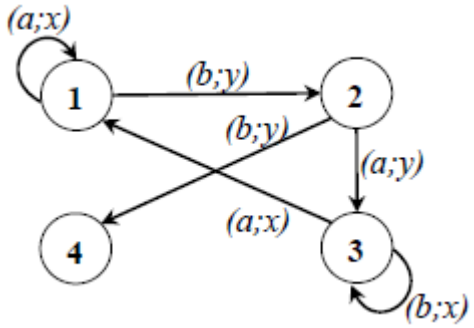
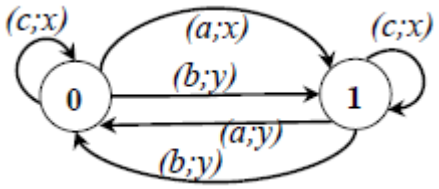
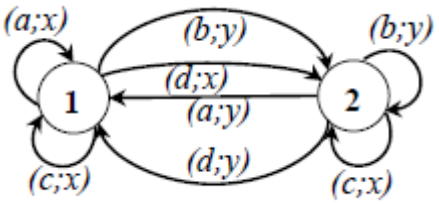
Задания:

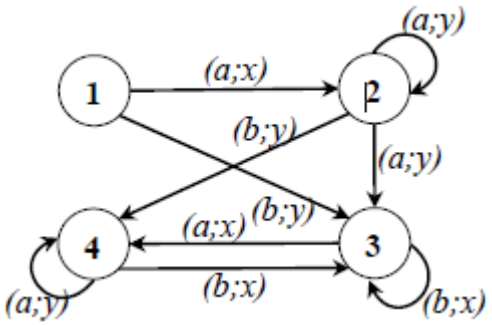
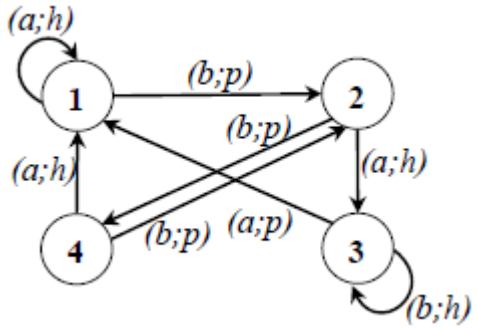
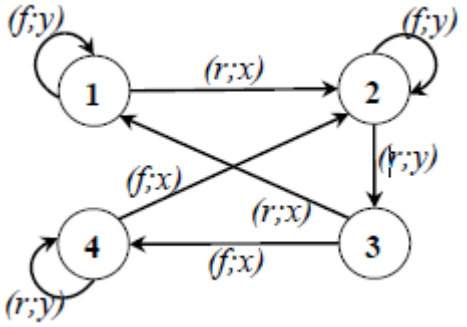
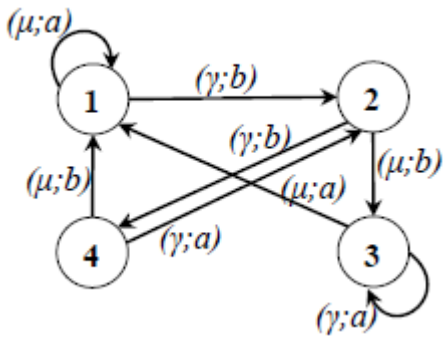
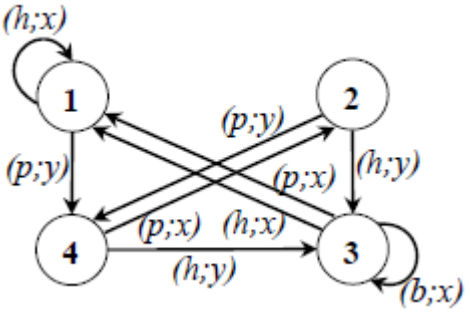
1. Для автоматов, заданных таблицами, построить диаграммы состояний:

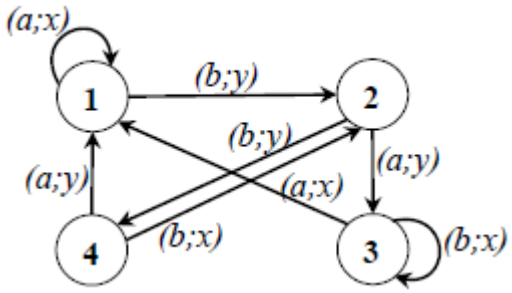
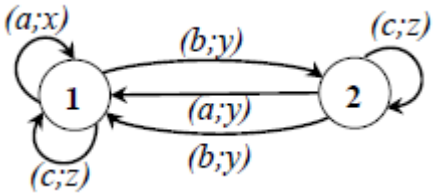
№ Варианта	Задание
1.	*U\Q* | *1* | *2* | *3* | *4* *a* | *1; x* | *3; y* | *1; x* | *1; y* *b* | *2; y* | *4; y* | *3; x* | *2; x*
2.	*U\Q* | *1* | *2* | *3* | *4* *c* | *3; x* | *1; y* | *2; x* | *2; y* *d* | *2; y* | *2; y* | *4; x* | *1; x*
3.	*U\Q* | *1* | *2* | *3* | *4* *k* | *2; z* | *1; y* | *1; z* | *2; y* *l* | *2; y* | *3; z* | *4; y* | *1; z*
4.	*U\Q* | *1* | *2* | *3* *a* | *3; x* | *3; y* | *1; x* *b* | *2; y* | *1; y* | *2; x*
5.	*U\Q* | *1* | *2* | *3* *c* | *1; a* | *3; b* | *1; b* *d* | *2; b* | *2; a* | *3; b*
6.	*U\Q* | *1* | *2* | *3* *k* | *1; s* | *3; f* | *1; s* *l* | *2; f* | *1; s* | *2; f*
7.	*U\Q* | *1* | *2* *a* | *1; x* | *1; y* *b* | *2; y* | *2; y* *c* | *1; x* | *2; x*

8.	<table><tr><td>*U\Q*</td><td>*1*</td><td>*2*</td></tr><tr><td>*x*</td><td>*1; x*</td><td>*2; y*</td></tr><tr><td>*y*</td><td>*2; y*</td><td>*1; y*</td></tr><tr><td>*z*</td><td>*1; x*</td><td>*2; x*</td></tr></table>
9.	<table><tr><td>*U\Q*</td><td>*1*</td><td>*2*</td></tr><tr><td>*a*</td><td>*1; x*</td><td>*1; y*</td></tr><tr><td>*b*</td><td>*2; y*</td><td>*2; y*</td></tr><tr><td>*c*</td><td>*1; x*</td><td>*2; x*</td></tr><tr><td>*d*</td><td>*2; x*</td><td>*1; y*</td></tr></table>
10.	<table><tr><td>*U\Q*</td><td>*0*</td><td>*1*</td><td>*2*</td><td>*3*</td></tr><tr><td>*a*</td><td>*1; x*</td><td>*3; y*</td><td>*0; x*</td><td>*1; y*</td></tr><tr><td>*b*</td><td>*2; x*</td><td>*0; x*</td><td>*3; y*</td><td>*2; x*</td></tr></table>

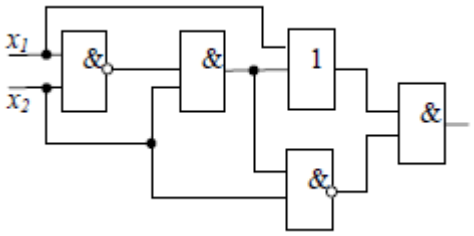
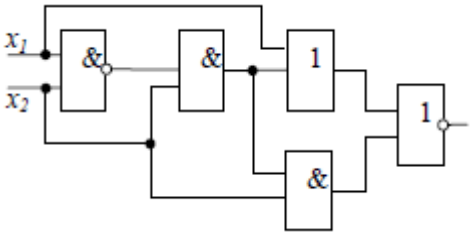
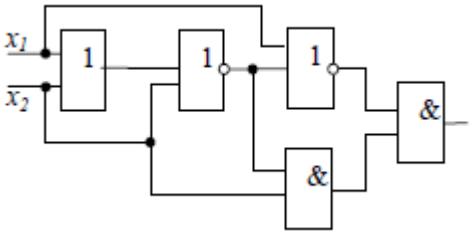
2. Задать табличным способом автомат, заданный диаграммой состояний:

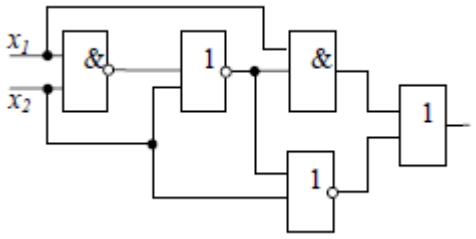
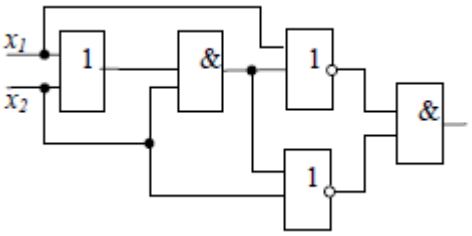
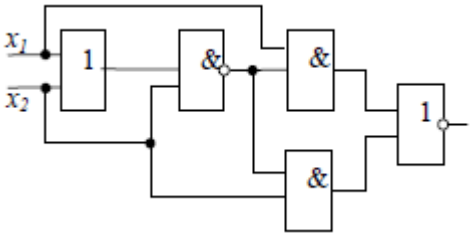
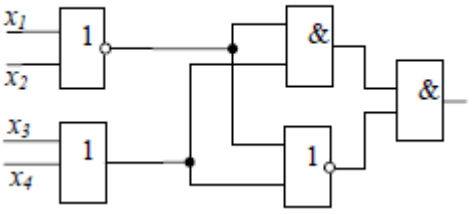
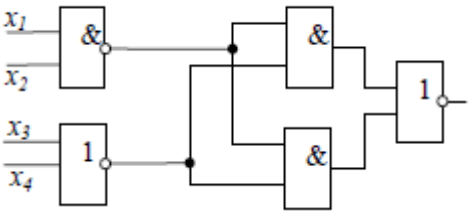
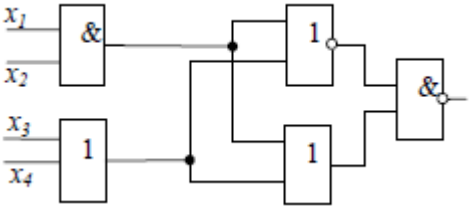
№ Варианта	Задание
1.	
2.	
3.	

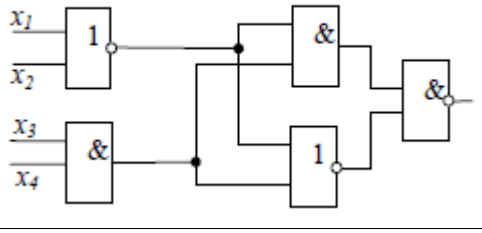
4.	 (a;y) (a;x) 1 2 (b;y) (a;y) (a;x) (b;y) 4 3 (b;x) (a;y) (b;x)
5.	 (a;h) (b;p) 1 2 (b;p) (a;h) (a;h) (b;p) (a;p) 4 3 (b;h)
6.	 (f;y) (f;y) (r;x) 1 2 (r;y) (f;x) (r;x) 4 3 (f;x) (r;y)
7.	 (μ;a) (γ;b) 1 2 (γ;b) (μ;b) (μ;b) (μ;a) 4 (γ;a) 3 (γ;a)
8.	 (h;x) 1 2 (p;y) (p;y) (p;x) (h;y) (p;x) (h;x) 4 3 (h;y) (b;x)

9.	
10.	

3. Для заданной комбинационной схемы построить аналитическое выражение, упростить его с помощью равносильных преобразований и нарисовать упрощенную схему:

№ Варианта	Задание
1.	
2.	
3.	

4.	 x_1 x_2 & 1 & 1 1
5.	 x_1 x_2 1 & 1 & 1
6.	 x_1 x_2 1 & & 1 &
7.	 x_1 x_2 x_3 x_4 1 1 & 1 &
8.	 x_1 x_2 x_3 x_4 & 1 & & 1
9.	 x_1 x_2 x_3 x_4 & 1 1 1 &

10.	

4. Для заданной логической таблицы функции F(a, b, c) записать аналитическое выражение и построить комбинационную схему:

№ Варианта	Задание

1.

a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

2.

a	b	c	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

3.

a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

4.

a	b	c	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

5.

a	b	c	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

6.

a	b	c	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

7.

a	b	c	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

8.

a	b	c	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

9.

a	b	c	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

10.

a	b	c	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Лабораторная работа № 2: Детерминированные автоматы. Эквивалентные состояния и автоматы

Цель работы: научиться находить эквивалентный детерминированный автомат, составлять определяющие таблицы и программы.

Детерминированный конечный автомат— набор из пяти элементов: $(\Sigma, Q, s \in Q, T \subset Q, \delta: Q \times \Sigma \to Q)$,
где Σ — алфавит, Q — множество состояний, s — начальное (стартовое) состояние, T — множество допускающих состояний, δ — функция переходов.

Детерминированный конечный автомат является специальным случаем недетерминированного конечного автомата, в котором:

• отсутствуют состояния, имеющие e-переходы;

• для каждого состояния s и входного символа a существует не более одной дуги, исходящей из s и помеченной как a.

Детерминированный конечный автомат имеет для любого входного символа не более одного перехода из каждого состояния. Если для представления функции переходов ДКА используется таблица, то каждая запись в ней представляет собой единственное состояние. Следовательно, очень просто проверить, допускает ли данный ДКА некоторую строку, поскольку имеется не более одного пути от стартового состояния, помеченного этой строкой. Следующий алгоритм имитирует поведение ДКА при обработке входной строки.

Состояния q автомата M и q' автомата M' считаются эквивалентными, если оба автомата, получив одну и ту же (любую) входную последовательность символов, перерабатывают ее в одинаковую выходную последовательность.

Автоматы M и M' называются эквивалентными, если для каждого состояния автомата M существует эквивалентное ему состояние автомата M' и наоборот.

Другими словами, эквивалентные автоматы реализуют одинаковые преобразования, но могут иметь различное число внутренних состояний.

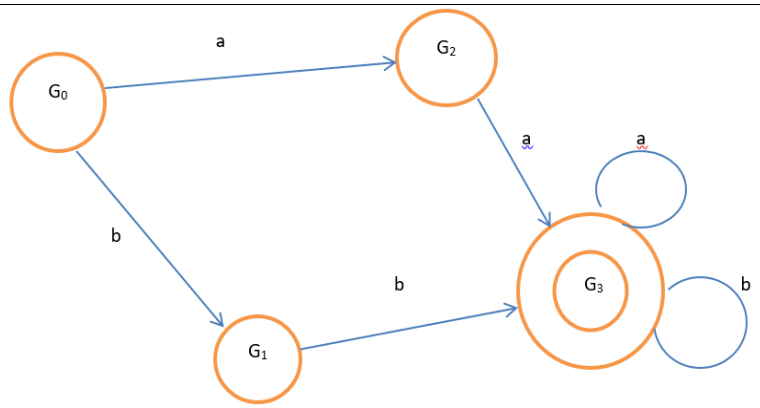
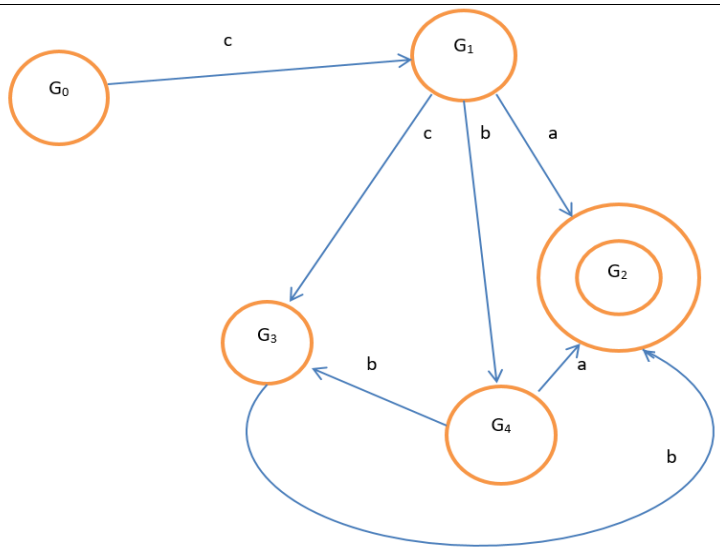
Понятие эквивалентности состояний применимо и к одному автомату (формально можно считать, что M и M' совпадают). Для одного автомата эквивалентными будут различные состояния, через которые одна и та же входная последовательность символов преобразуется в одинаковую выходную.

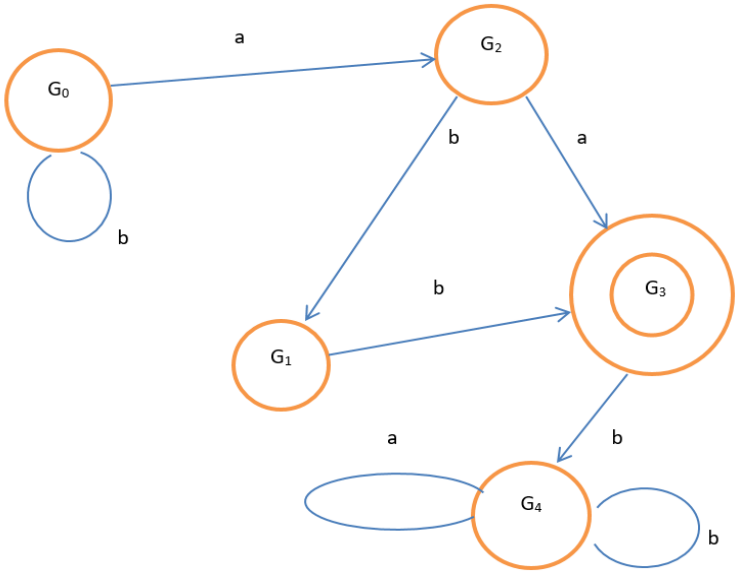
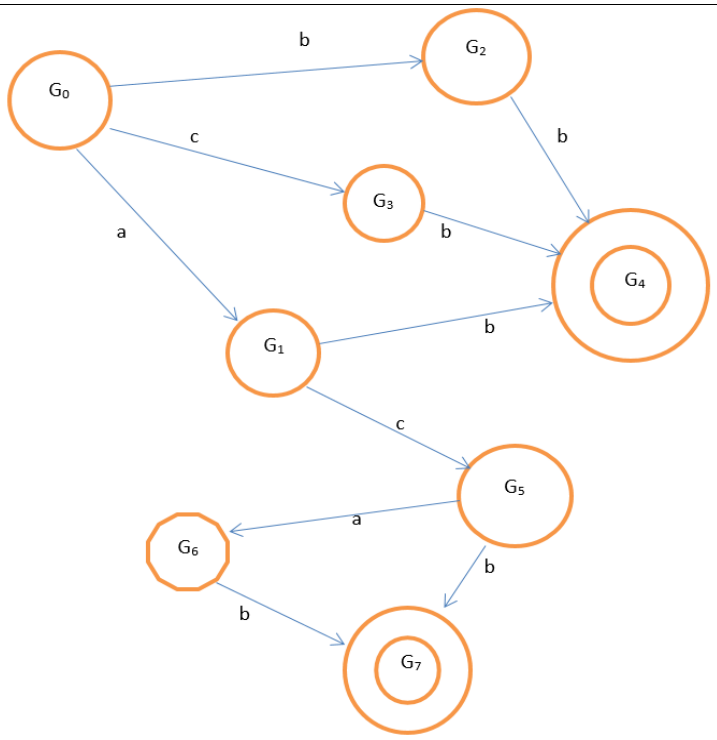
Вопросы для самоконтроля:

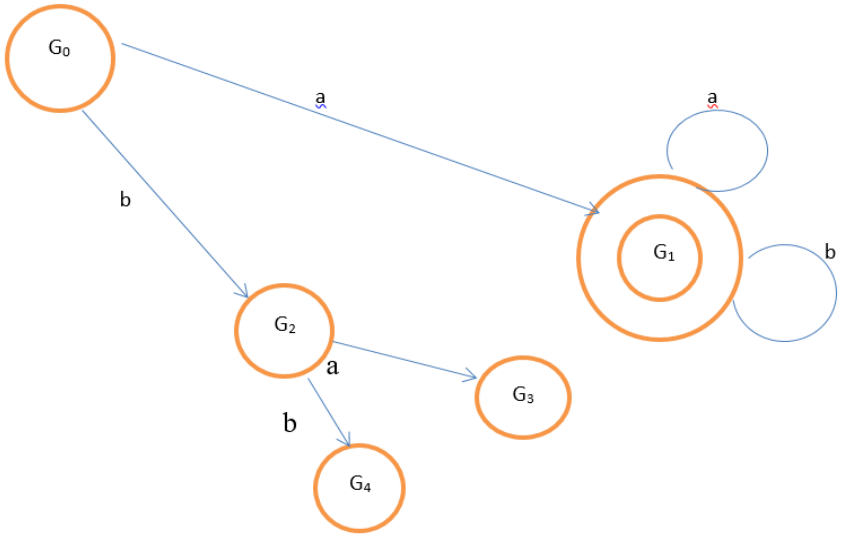
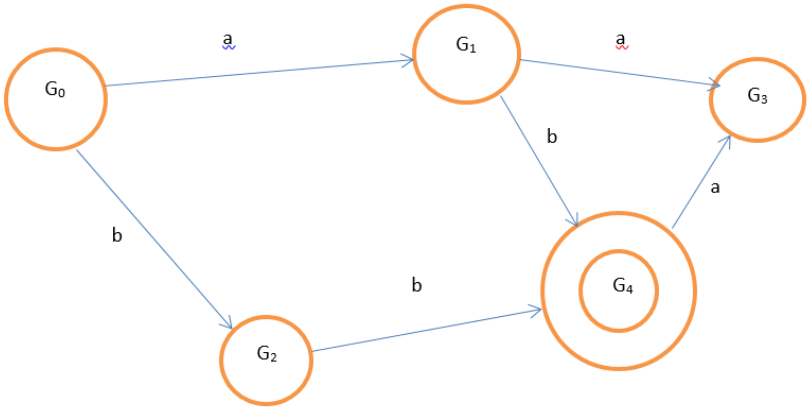
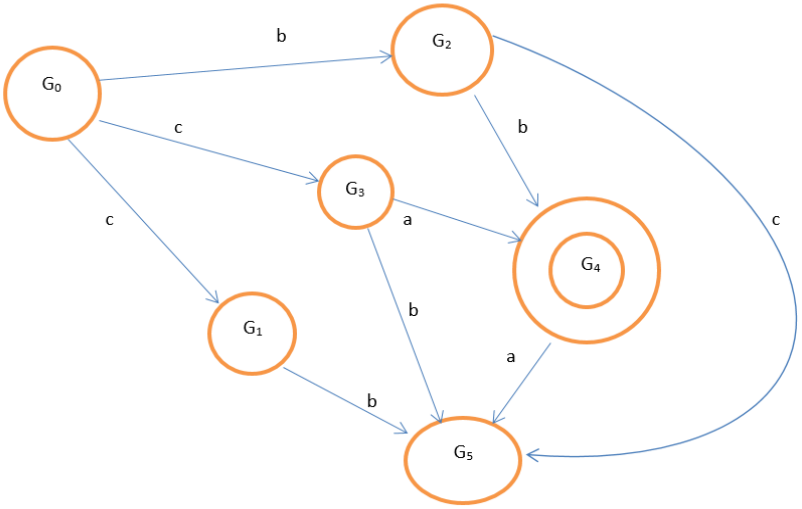
1) Дайте определение детерминированного конечного автомата.
2) В каких случаях автомат становится детерминированным?
3) Какие автоматы являются эквивалентными?
4) Какие состояния автомата являются эквивалентными?

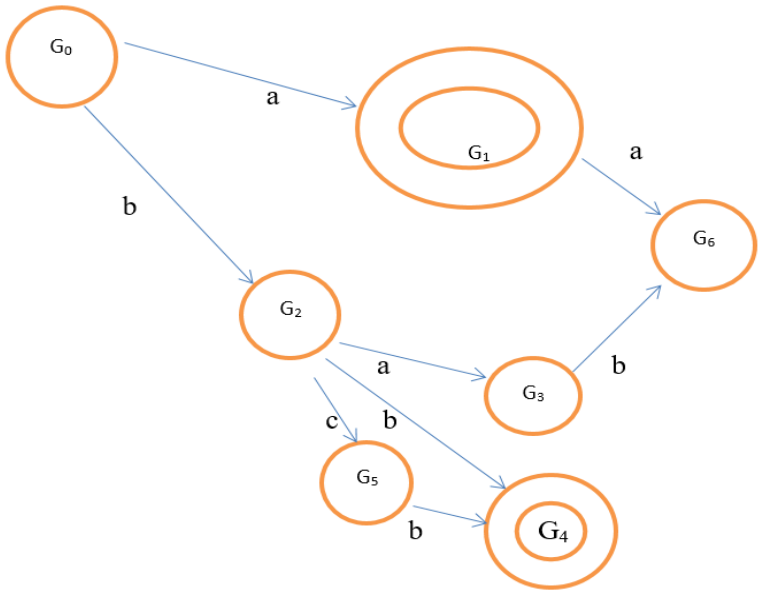
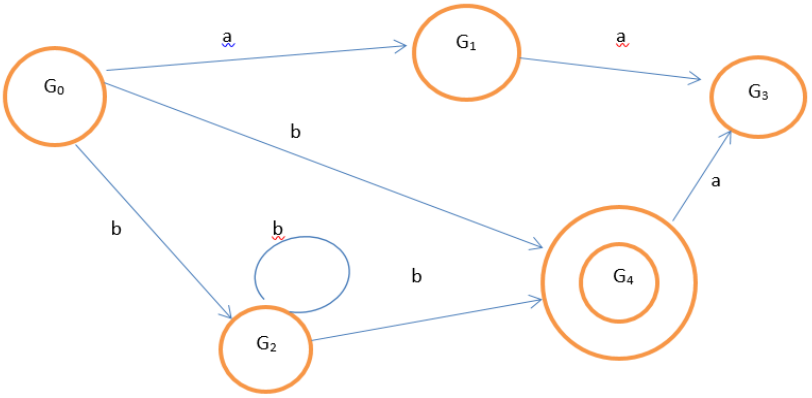
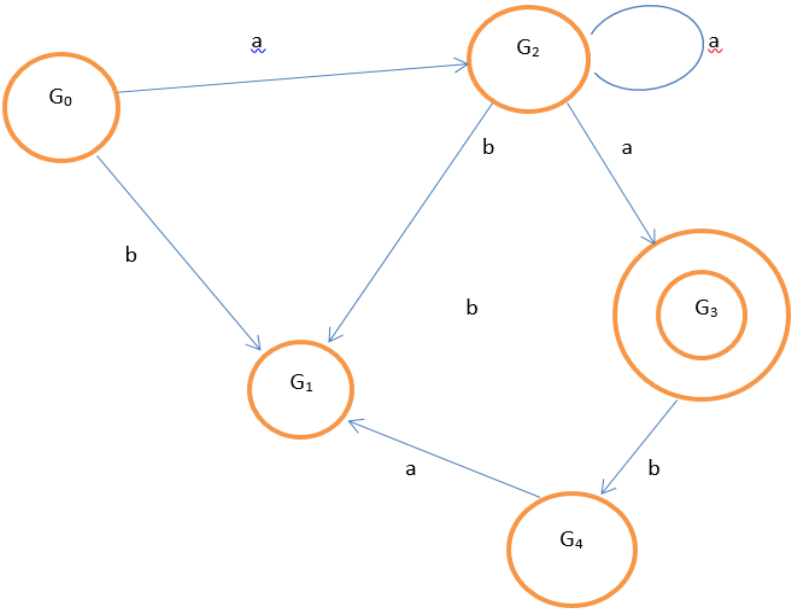
Задания:

1. Для данного недетерминированного автомата найти эквивалентный ему детерминированный автомат:

№ Варианта	Задание
1.	
2.	

3.	 a G0 G2 b a b G3 b G1 a b G4 b
4.	 b G0 G2 c b G3 a b G4 G1 b c G5 a G6 b b G7

5.	 G0 a a b G1 b G2 a G3 b G4
6.	 a G1 a G0 G3 b a b G4 b G2
7.	 b G2 G0 b c G3 c a G4 c b G1 a b G5

8.	 G0 a G1 a b G6 G2 a G3 b c b G5 b G4
9.	 a G1 a G0 G3 b a b b G4 b G2
10.	 a G2 a G0 b a b G3 b G1 a b G4

2. Какой язык допускает детерминированный автомат, полученный в задании 1.

3. Постройте детерминированный автомат, для которого указанный язык является допустимым (в алфавите {a,b,c}):

№ Варианта	Задание
1.	Множество слов, начинающихся с подслов aba или bab
2.	Множество слов, в которых встречаются последовательности aba или bab
3.	Множество слов, в которых встречаются последовательности aa или bb, или cc
4.	Множество слов, в которых имеется подслово abc
5.	Множество слов, начинающихся с последовательности abc
6.	Множество слов, оканчивающихся буквой a
7.	Множество слов, оканчивающихся сочетанием ba
8.	Множество слов, оканчивающихся сочетанием bac
9.	Множество слов, оканчивающихся сочетанием cba
10.	Множество слов, оканчивающихся буквой b

4. Для автомата задания 3 составить определяющие таблицы.

5. Для автоматов (недетерминированного и детерминированного) задания 1 записать определяющие таблицы и программы.

Лабораторная работа № 3: Вычисление функций с помощью машины Тьюринга.

Цель работы: научиться вычислять функции с помощью машины Тьюринга, составлять блок-схемы работы машины Тьюринга.

Машина Тьюринга – абстрактная вычислительная машина, предложенная Аланом Тьюрингом в 1936 году для формализации понятия алгоритма.

Машина Тьюринга состоит из ленты, управляющего устройства и считывающей головки.

Лента разбита на ячейки. Во всякой ячейке в каждый момент времени находится в точности один символ из внешнего алфавита $A = \{a_0, a_1, \dots, a_{n-1}\}, n \geq 2$. Некоторый символ алфавита A называется пустым, любая ячейка, содержащая в данный момент пустой символ, называется пустой ячейкой.

В дальнейшем в качестве внешнего алфавита будем использовать $A = \{0, 1\}$, где в качестве пустого символа будем использовать 0 (нуль). В каждый момент времени лента содержит конечное число ячеек, но в процессе работы машины можно пристраивать новые ячейки в пустом состоянии.

Управляющее устройство в каждый момент времени находит в некотором состоянии q_i, принадлежащее множеству $Q = \{q_0, q_1, \dots, q_{r-1}\}, r \geq 1$. Множество Q называется внутренним алфавитом. В дальнейшем начальное состояние будем обозначать символом q_1, а заключительное символом q_0 .

Считывающая головка перемещается вдоль ленты так, что в каждый момент времени она обозревает ровно одну ячейку ленты. Головка может считывать содержимое обозреваемой ячейки и записывать в нее вместо обозреваемого символа некоторый новый символ из внешнего алфавита.

Работа машины Тьюринга определяется программой. Программа состоит из команд. Каждая команда представляет собой выражение одного из следующего вида:

1) $q_i a_j \rightarrow q_k a_e$;
2) $q_i a_j \rightarrow q_k a_e R$;
3) $q_i a_j \rightarrow q_k a_e L$.

Команда 1 заключается в том, что содержимое a_j обозреваемой на ленте ячейки стирается, а на его место дописывается символ a_e

(который может совпадать с a_j), машина переходит в новое состояние q_k (оно может совпадать с предыдущим состоянием q_i).

Команда 2 работает аналогично команде 1, и дополнительно сдвигает считывающую головку в соседнюю справа ячейку.

Команда 3 работает аналогично команде 1, и дополнительно сдвигает считывающую головку в соседнюю слева ячейку.

Если считывающая головка находится в крайней справа (слева) ячейки ленты и происходит ее сдвиг вправо (влево), то к ленте пристраивается новая ячейка в пустом состоянии.

Вопросы для самоконтроля:

1) Дайте определение машины Тьюринга.
2) Из чего состоит машина Тьюринга?
3) Какими командами определяется программа машины Тьюринга?

Задания:

1. Какую функцию вычисляет данная программа машины Тьюринга? Составить блок-схему работы этой машины, считая ее одноместной:

№ варианта.	Программа машины Тьюринга.
1	{$q_0 1\ q_0 0R$, $q_0 0\ q_0 1L$, $q_1 0\ q_2 1R$}
2	{$q_0 1\ q_0 0R$, $q_1 0\ q_1 1R$, $q_1 1\ q_1 0R$}
3	{$q_0 1\ q_0 0R$, $q_0 0\ q_1 1L$, $q_1 0\ q_1 0R$, $q_1 1\ q_1 1L$}
4	{$q_0 1\ q_0 1L$, $q_0 0\ q_1 1L$, $q_1 0\ q_1 1E$}
5	{$q_0 1\ q_0 0R$, $q_0 0\ q_1 0L$, $q_1 0\ q_1 0L$}
6	{$q_0 1\ q_0 0R$, $q_0 0\ q_1 0L$, $q_1 0\ q_2 0L$}
7	{$q_0 1\ q_0 0R$, $q_0 0\ q_1 1R$, $q_1 0\ q_2 1R$, $q_2\ 0q_3\ 1L$, $q_3\ 1q_3\ 1L$, $q_3\ 0q_4\ 0R$}
8	{$q_0 1\ q_0 1R$, $q_0 0\ q_1 0R$, $q_1 1\ q_2 0L$, $q_2\ 1q_2\ 1L$, $q_2\ 0q_3\ 0R$}
9	$\{q_0 1 q_0 1R, q_0 0 q_0 0L\}$
10	$\{q_0 1 q_0 0R, q_0 0 q_0 1L, q_0 1 q_{rîä} 1E\}$

2. Написать программу машины Тьюринга для вычисления указанной функции. Составить блок-схему ее работы:

№ варианта.	Функция.
1	$f(x, y) = x + 2$
2	$f(x, y) = x - y$
3	$f(x, y) = x - y + 2$
4	$f(x, y) = x + y + 4$
5	$f(x, y) = y + 3$
6	$f(x, y) = y - x$
7	$f(x, y) = y - x - 2$
8	$f(x, y) = x + y - 4$
9	$f(x) = 5$
10	$f(x, y) = 3$

Лабораторная работа № 4: Рекурсивные функций на примере машины Тьюринга.

Цель работы: научиться доказывать рекурсивные функции с помощью машины Тьюринга.

Систему, эквивалентную машине Тьюринга, можно построить на основе математических функций. Для этого, нам требуется ввести следующие классы функций: примитивно рекурсивные функции, общерекурсивные функции, частично рекурсивные функции.

Последний класс будет совпадать с классом вычислимых по Тьюрингу функций (то есть функций, для вычисления которых можно построить алгоритм для машины Тьюринга).

Определение алгоритма через рекурсивные функции по сути лежит в основе лямбда-исчисления, и на его основе строится подход функционального программирования.

Класс примитивно рекурсивных функций содержит базовые функции и все функции, получающиеся из базовых посредством операторов подстановки и примитивной рекурсии.

К базовым функциям относятся:

- Нулевая функция $O() = 0$ – функция без аргументов, которая всегда возвращает 0;
- Функция следования $S(x) = x + 1$ – функция, которая любому натуральному числу x ставит в соответствие следующее число $x + 1$;
- Функции $I_n^m(x_1, \dots, x_n) = x_m$, где $0 < m \leq n$ – функции от n переменных, которые любому набору натуральных чисел $x_1, \dots, x_n$ ставят в соответствие число x_m из этого набора.

Для конструирования остальных функций класса используются операторы:

1) Подстановки. Для функции f от m переменных и m функций $g_1, \dots, g_m$ от n переменных каждая, результатом подстановки g_k в f является функция $h(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_m(x_1, \dots, x_n))$ от n переменных;
2) Примитивной рекурсии. Пусть $f(x_1, \dots, x_n)$ – функция от n переменных, а $g(x_1, \dots, x_{n+2})$ – функция от $n + 2$ переменных. Тогда результатом применения оператора примитивной рекурсии к функциям f и g является функция h от $n + 1$ переменной вида:

$$h(x_1, \dots, x_n, 0) = f(x_1, \dots, x_n)$$
$$h(x_1, \dots, x_n, y + 1) = g(x_1, \dots, x_n, y, h(x_1, \dots, x_n, y))$$

Класс частично рекурсивных функций включает примитивно рекурсивные функции, и, плюс к этому, все функции, которые получаются из примитивно рекурсивных с помощью оператора минимизации аргумента:

Пусть f – функция от n переменных $x_i \in N$. Тогда результатом применения оператора минимизации аргумента к функции f является функция h от $n-1$ аргумента, определяемая как:

$$h(x_1, \dots, x_{n-1}) = \min y,$$

где

$$f(x_1, \dots, x_{n-1}, y) = 0.$$

То есть, h возвращает минимальное значение последнего аргумента функции f при котором значение f равно нулю.

В то время как примитивно рекурсивные функции всегда вычислимы, частично рекурсивные функции при некоторых значениях аргументов могут быть не определены.

Более строго частично рекурсивные функции следовало бы называть «частично определенные рекурсивные функции», поскольку они определены только на части возможных значений аргументов.

Общерекурсивные функции – это подкласс всех частично рекурсивных функций, которые определены для любых значений аргументов. Задача определения того, является ли данная частично рекурсивная функция общерекурсивной является алгоритмически неразрешимой.

Вопросы для самоконтроля:

1) Дайте определение примитивно рекурсивной функции.
2) Дайте определение частично рекурсивной функции.
3) Что такое оператор минимизации аргумента?
4) Дайте определение общерекурсивной функции.

Задания:

1. Описать блок-схему машины Тьюринга, вычисляющую данную функцию:

№ варианта.	Функция.
1	$f(x, y) = x! - y + 2$
2	$f(x, y) = x \times y + 2$

3	$f(x, y) = x^y + 2$
4	$f(x, y) = y^x + 4$
5	$f(x, y) = y + 3^x$
6	$f(x, y) = y - x^2$
7	$f(x) = 5 + x!$
8	$f(x, y) = x + [\log_2 y]$
9	$f(x, y) = [\log_2 y] + 2$
10	$f(x, y) = 41 + x \times y^2$

2. Доказать рекурсивность функции:

№ варианта.	Множество
1	$x + y \div k$
2	$x \cdot y + 1$
3	$x \cdot y \div x$
4	$x^y + k$
5	$x \cdot y + [log_x y]$
6	$x \div [log_x y]$
7	$(x + y)^2 + 3 \cdot x + y$
8	$(x + y)^2 \div y$
9	$[log_x y] + [log_y x]$
10	$e^x + y$

3. Доказать общую рекурсивность функции:

№ варианта.	Множество
1	$\left[\frac{x}{y}\right]$ – целая част от деления, причем $\left[\frac{x}{0}\right] = x.$
2	$res(x, y)$ –остаток от деления x на y, причем $res(x, 0) = x.$
3	$\tau(x)$-число делителей числа x, причем $\tau(0) = 0.$
4	$\sigma(x)$-сумма делителей числа x, причем $\sigma(0) = 0.$
5	$Ih(x)$-число простых делителей числа x, причем $Ih(0) = 0.$
6	$\pi(x)$-число простых чисел, не превосходящих числа x.
7	$P(x)$ – x-е простое число, причем $P(0) = 2$, $P(1) = 3$ и т.д.
8	$long(x)$ – номер наибольшего простого делителя числа x.
9	$ex(x, y)$ – показатель x-го простого числа в каноническом разложении на простые множители числа y , причем $ex(x, 0) = 0.$
10	$\left[\sqrt{x}\right]$ – целая часть квадратного корня из числа x.

Лабораторная работа № 5: Характеристики сложности алгоритма для машины Тьюринга.

Цель работы: научиться определять различные характеристики сложности алгоритма для машины Тьюринга.

Различные подходы к уточнению понятия «алгоритм» позволяли изучать принципиальную возможность решения некоторой математической задачи. Однако теоретическая возможность алгоритмического решения задачи еще не гарантирует практическую реализуемость алгоритма. Поэтому необходимо ввести характеристики алгоритмов, которые бы показывали степень практической реализуемости алгоритмов. Другая причина, по которой необходимы такие характеристики, – необходимость сравнения эффективности различных алгоритмов, которые решают одну и ту же задачу.

При решении некоторой задачи P алгоритмом A обычно рассматривают такие характеристики:

1) количество шагов $T_a(x)$, которое необходимо сделать алгоритму A для получения результата при использовании входных данных x. Величина $T(A,n) = \max\limits_{|x|=n} T_a(x)$ (максимум берется по всем входным данным объема n) называется *временной сложностью* алгоритма A.

2) объем памяти $M_a(x)$,, необходимый для хранения всех входных и промежуточных данных в процессе выполнения алгоритма при использовании входных данных x. Величина $M(A,n) = \max\limits_{|x|=n} M_a(x)$ (максимум берется по всем входным данным объема n) называется *емкостной сложностью* алгоритма A.

Для определения временной сложности алгоритма вместо общего числа шагов алгоритма можно также использовать количество операций определенного вида. Аналогично можно определить средние величины временной и емкостной сложности алгоритма. Сложность задачи P – это сложность наилучшего алгоритма, известного для ее решения, т.е. $S(P,n) = \min\limits_{A} T(A,n)$ (минимум берется по всем алгоритмам для задачи P).

Задача называется N-задача, если ее можно полиномиально преобразовать в любую данную NP-задачу. Если можно решить полную NP -задачу, то можно решить и все NP-задачи. Класс NP охватывает многие задачи: задача о выполнимости, задача коммивояжера, решение систем уравнений с целыми переменными, составление расписаний с определенными условиями, задача о рюкзаке, оптимальный раскрой и

т.д. Все они решаются на детерминированных машинах Тьюринга экспоненциально. Они трудны, но не доказано, что их нельзя упростить. Если хотя бы одну из них можно решить полиномиально, то все другие также решаются полиномиально. Общие подзадачи N-задач могут быть легкоразрешимыми.

Вопросы для самоконтроля:

1) Какие существуют характеристики сложности алгоритмов?
2) Дайте определение временной сложности алгоритма.
3) Дайте определение емкостной сложности алгоритма.
4) Дайте определение N-задачи.

Задания:

1. Определить необходимую память (число ячеек на ленте) для машины Тьюринга, вычисляющую функцию:

№ варианта.	Множество
1	$x + k$
2	k
3	$x + y$
4	$x \cdot y$
5	$x \div y$
6	проверить равенство $x = y$
7	определить четность числа x
8	$\max(x, y)$
9	$\min(x, y)$
10	$x + y \div k$

2. Определить необходимую память (число ячеек на ленте) и временную сложность (число тактов работы) для машины Тьюринга, вычисляющую функцию:

№ варианта.	Множество
1	$q_0 1001^x 0 \rightarrow q_k 101^x 00$ (перенос 0 вправо)
2	$q_0 1001^x 0 \rightarrow 0 q_k 101^x 0$ (перенос 0 влево)
3	$q_0 1^y 01^x 0 \rightarrow q_k 1^x 01^y 0$ (поменять местами последовательности)

4	$q_0 1^y 01^x 0 \to \begin{cases} 1, \text{если } y \text{ больше } x; \\ 0, \text{в противном случае.} \end{cases}$
5	$q_0 1^y 01^x 0 \to \begin{cases} 1, \text{если } y \text{ меньше } x; \\ 0, \text{в противном случае.} \end{cases}$
6	$q_0 1^y 01^x 0 \to \begin{cases} q_0 1^{y-x}, \text{если } y \text{ больше } x; \\ 0, \text{в противном случае.} \end{cases}$
7	$q_0 1^y 01^x 0 \to \begin{cases} q_0 1^{x-y}, \text{если } y \text{ меньше } x; \\ 0, \text{в противном случае.} \end{cases}$
8	$q_0 1^y 01^x 0 \to 0 q_k 1^{x+y} 0$
9	$q_0 1^y 01^x 01^z 0 \to 0 q_k 1^{x+y+z} 0$
10	$q_0 1^y 01^x 0 \to 0 q_k 1^{x+y \div k} 0$, где k постоянное число.

3. Определить временную сложность (число тактов работы) машины Тьюринга, вычисляющую функцию:

№ варианта.	Функция.
1	$x + y \div k$
2	$\min(x, y)$
3	$\max(x, y)$
4	определить четность числа x
5	проверить равенство $x = y$
6	$x \div y$
7	x- четно
8	$x + y$
9	k
10	$x + k$

4. Привести пример NP-полной задачи.

Лабораторная работа № 6: Сети Петри

Задание на лабораторную работу:

Диспетчер управляет внутризаводским транспортом и имеет в своем распоряжении два грузовика. Заявки на перевозки поступают к диспетчеру каждые τ_1 мин. С вероятностью P_1 диспетчер запрашивает по радио один из грузовиков и передает ему заявку, если тот свободен. В противном случае он запрашивает другой грузовик и таким образом продолжает сеансы связи, пока один из грузовиков не освободится. Каждый сеанс связи длится ровно τ мин. Диспетчер допускает накопление у себя до пяти заявок, после чего вновь прибывшие заявки получают отказ. Грузовики выполняют заявки на перевозку за τ_2 мин.

Смоделировать работу внутризаводского транспорта в течение T часов. Подсчитать число обслуженных и отклоненных заявок. Определить коэффициенты загрузки грузовиков.

Постановка задачи:

Диспетчер управляет транспортом и имеет в распоряжении два грузовика, на которые поступают заявки. Интервал времени между двумя заявками - τ_1 распределен по равномерному закону в интервале $\tau_1 = a_1 \pm b_1$. Вероятность запроса диспетчером грузовика - P_1, и, если тот свободен, то принимает заявку, в противном случае запрашивается другой грузовик. Это сеанс связи, который длится ровно τ мин.

Максимальное количество заявок $N_1 = 5$. Если $N_1 > 5$, то последующие заявки не рассматриваются.

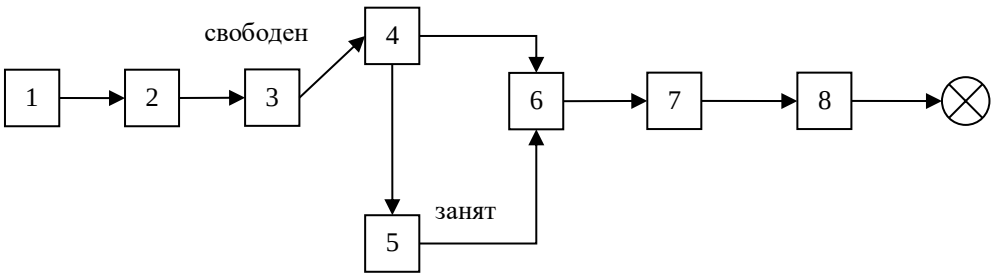
Интервал времени выполнения заявки грузовиком τ_2 распределен по равномерному закону в интервале $\tau_2 = a_2 \pm b_2$.

Требуется смоделировать работу грузовиков в течение T часов ($T \neq 0$), считая, что первая заявка поступает в момент времени равный 0.

В результате моделирования требуется определить следующие характеристики:

1. Количество и номера заявок, завершивших обслуживание;
2. Количество и номера заявок, получивших отказ;
3. Коэффициенты загрузки грузовиков k_1 и k_2, которые соответственно равны: $k_1 = \frac{t_{раб.1}}{t_{общ}}$ и $k_2 = \frac{t_{раб.2}}{t_{общ}}$.

Принципиальная схема работы внутризаводского транспорта:



- Блок 1 – формирование входного массива грузовиков на основе исходного массива моментов времени поступления заявок на грузовики.
- Блок 2 – получение заявки.
- Блок 3 – обработка информации о количестве заявок.
- Блок 4 – запрос первого грузовика.
- Блок 5 – запрос второго грузовика.
- Блок 6 – установление связи с грузовиком.
- Блок 7 – выполнение заявки.
- Блок 8 – логический блок, определяющий переход к рассмотрению очередной заявки или завершение процесса анализа обработки заявок.

Вариант задания:

№	ф	a_1	b_1	a_2	b_2	P_1	T
1	3	5	3	60	10	0.3	
2	3	6	3	70	20	0.4	
3	3	7	3	60	30	0.5	
4	3	8	3	70	10	0.3	
5	3	7	3	80	20	0.4	
6	3	6	3	90	30	0.5	
7	3	5	3	40	10	0.3	
8	3	6	3	50	20	0.4	
9	3	7	3	60	30	0.5	
10	3	9	3	50	10	0.3	
11	3	8	3	40	20	0.4	
12	3	7	3	60	30	0.5	
13	3	6	3	70	10	0.3	10
14	3	5	3	60	20	0.4	
15	3	8	3	70	30	0.5	
16	3	9	3	80	10	0.3	
17	3	5	3	90	20	0.4	
18	3	6	3	40	30	0.5	
19	3	7	3	50	10	0.3	
20	3	8	3	60	20	0.4	
21	3	7	3	50	30	0.5	
22	3	6	3	40	10	0.3	
23	3	5	3	60	20	0.4	
24	3	6	3	70	30	0.5	
25	3	7	3	60	10	0.3	

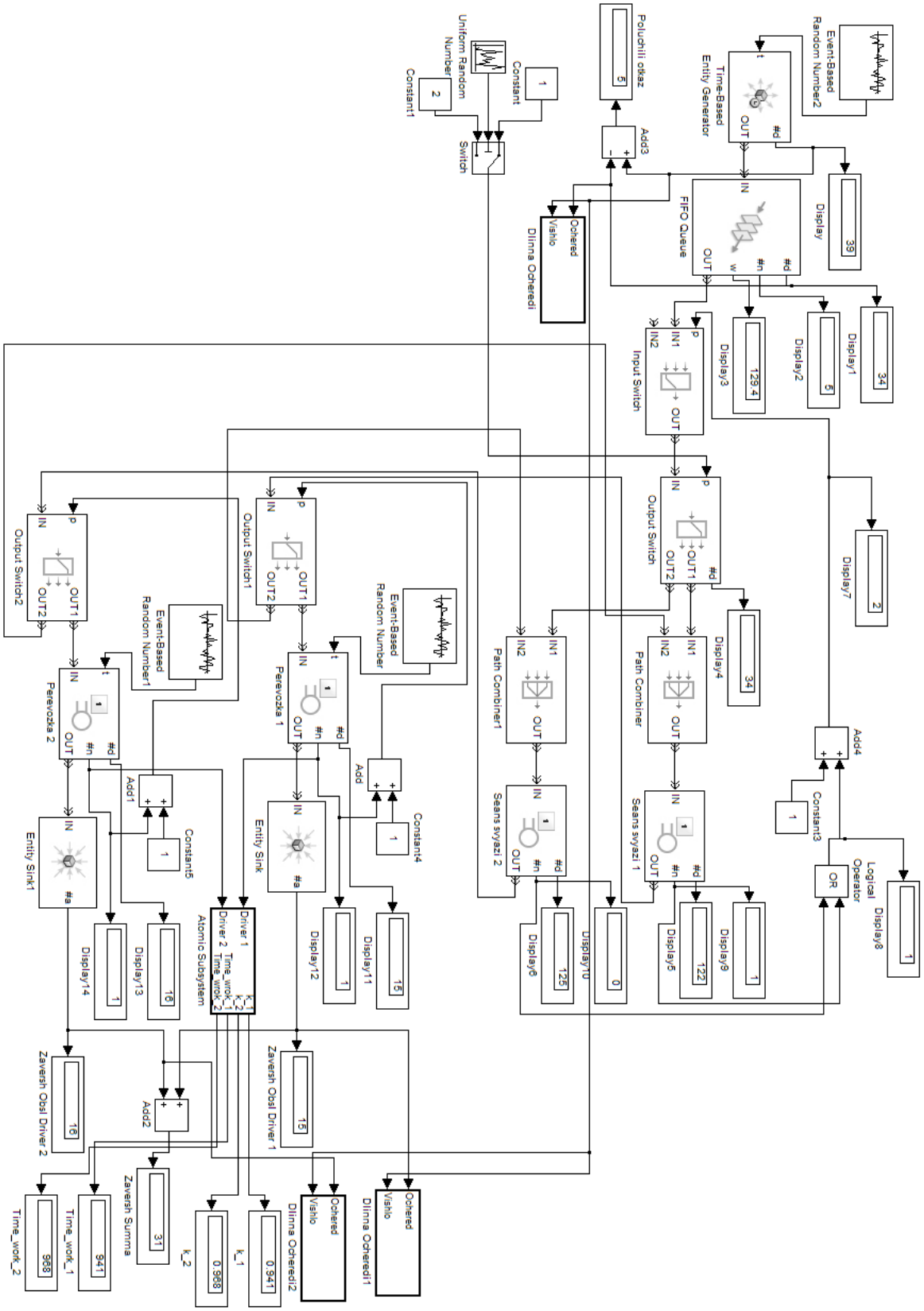
Пример реализации:

В процессе создания схемы были использованы следующие блоки библиотеки Sim Events:

- Time based Entity Generator - блок формирования распределенных во времени сигналов, имитирующих последовательность поступающих на вход системы запросов на обслуживание
- Event-Based random Number - блок формирования временных интервалов, используемых в качестве времен обслуживания заявок
- Path Combiner – блок для объединения входных сигналов
- Input Switch – управляемый ключ
- Output Switch – управляемый ключ
- FIFO Queue - блок, реализующий очередь обслуживания заявок

- Single Server – обслуживающий блок
- Entity Sink – приемник обслуженных заявок

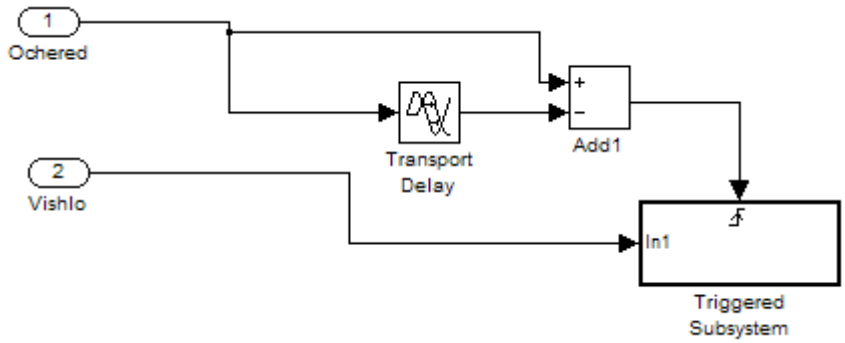
Сама схема выглядит следующим образом:



Пример реализации подсистем:

Подсистема Dlinna Ocheredi:

Структура подсистемы:



Подсистемы осуществляют запись в рабочую облать номеров заявок, завершивших обслуживание и получивших отказ. Для записи номеров заявок, получивших отказ, на вход 1 подается параметр статистики #d блока FIFO Queue (очередь заявок), а на вход 2 подается параметр статистики #d блока Time-Based Entity Generator (количество сгенерированных заявок).

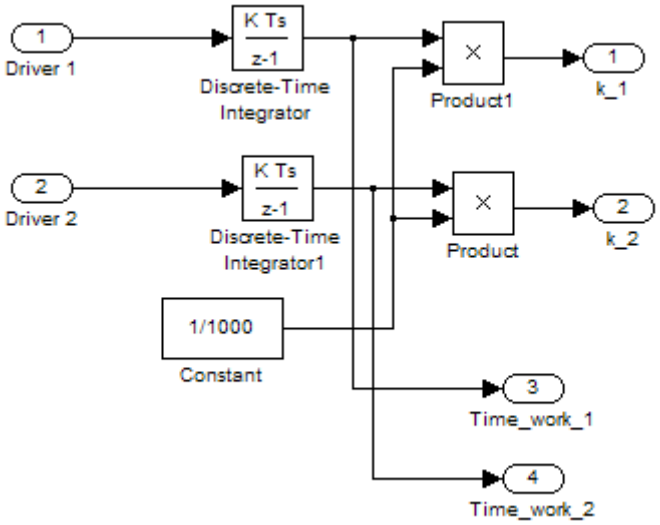
Для записи номеров заявок, завершивших обслуживание, на вход 1 подается парметр статистики #a блока Entity Sink (количество выполненых заявок, отдельно для каждого грузовика), а на вход 2 подается параметр статистики #d блока Time-Based Entity Generator (количество сгенерированных заявок).

Триггерная подсистема осуществляет саму запись в рабочую область.

Параметры блока Transport Delay (необходим для получения прямоугольных импульсов для включения триггерной подсистемы):

Подсистема Work:

Структура подсистемы:



Подсистема осуществляет вычисление коэффициентов загрузки грузовиков k_1 и k_2. На входа 1 и 2 подсистемы подаются соответственно значения параметров статистики #n блоков Perevozka1 и Perevozka2 (текущее состояние блока, 1 или 0). На выходах 1 и 2 получаем время работы первого и второго грузовиков, а на выходах 3 и 4 получаем значения коэффициентов k_1 и k_2.

Вопросы для самоконтроля:

1) Что такое СП и с помощью каких параметров она задается?
2) Как выглядит уравнение состояния СП?
3) В чем заключаются матричные методы исследования СП-моделей?
4) Что такое полная *p*-цепь и полная *t*-цепь?

Библиографический список

1. Хаггарти, Р. Дискретная математика для программистов [Электронный ресурс] : учебное пособие / Р. Хаггарти ; пер. англ. под ред. С.А. Кулешов ; пер. с англ. А.А. Ковалев, В.А. Головешкин, М.В. Ульянов. - Изд. 2-е, испр. - М. :Техносфера, 2012. - 400 с. - Режим доступа : biblioclub.ru
2. Судоплатов, С. В. Дискретная математика [Электронный ресурс]: учебник / С. В. Судоплатов, Е. В. Овчинникова. - 4-е изд. - Новосибирск : НГТУ, 2012. - 278 с. – Режим доступа : biblioclub.ru
3. Дискретная математика [Электронный ресурс] : учебное пособие / Ю. Ю. Громов [и др]. - Тамбов : ТГТУ, 2012. - 128 с. - Режим доступа: biblioclub.ru
4. Шевелев, Ю. П. Дискретная математика [Текст] : учебное пособие/ Ю.П. Шевелев. - СПб. : Лань, 2008. – 592 с.
5. Новиков, Федор Александрович. Дискретная математика для программистов [Текст] : учебник для магистров и бакалавров / Ф. А. Новиков. – СПб.[и др.] : Питер, 2011. – 384 с.
6. Лавров И. А. Задачи по теории множеств, математической логике и теории алгоритмов [Текст] / И. А. Лавров, Л. Л. Максимова. -2-е изд. - М. : Наука, 1984. – 223 с.
7. Хаггарти, Р. Дискретная математика для программистов [Текст]: учеб.пособие/ Р. Хаггарти; пер. с англ. под ред. С. А. Кулешова. – М. : Техносфера, 2005. – 400 с.
8. Милых, В. А. Дискретная математика [Текст] : учебное пособие/ В. И. Милых, И. Г. Уразбахтин. – Курск: Курск ГТУ, 2006. – 139 с.
9. Палий, И. А. Дискретная математика [Текст] : курс лекций / И. А. Палий – М. : Эксмо, 2008. – 352с.
10. Аляев Ю. А. Дискретная математика и математическая логика [Текст] / Ю. А. Аляев, С.Ф. Тюрин– М. : Финансы и статистика, 2006. – 368 с.
11. Плотников А. Д. Дискретная математика [Текст] / А. Д. Плотников - М. : Новые знание, 2005.– 288 с.
12. Яблонский С. В. Введение в дискретную математику [Текст] : учебное пособие для вузов/ С. В. Яблонский. – 4-е изд., стер. - М. : Высш. школа. – 2003. - 384 с.