

**ГОСУДАРСТВЕННОЕ ВЫСШЕЕ УЧЕБНОЕ ЗАВЕДЕНИЕ
ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ
К ЛАБОРАТОРНЫМ РАБОТАМ
ПО КУРСУ “КОМПЬЮТЕРНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА”
Часть2**

**ГОСУДАРСТВЕННОЕ ВЫСШЕЕ УЧЕБНОЕ ЗАВЕДЕНИЕ
ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ И ЗАДАНИЯ
К ЛАБОРАТОРНЫМ РАБОТАМ
ПО КУРСУ “КОМПЬЮТЕРНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА”
Часть 2**

**(для студентов, обучающихся по направлению
“Программная инженерия”)**

Рассмотрено на заседании кафедры
Программная инженерия
Протокол №1 от 30.08.2016 г.

Утверждено на заседании
учебно-издательского совета ДонНТУ
Протокол № от ____ . ____ . ____ г.

УДК 004.021

Методические указания и задания к лабораторным работам по курсу "Компьютерная дискретной математики, часть 2" (для студентов, обучающихся по направлениям подготовки «Программная инженерия») / Сост.: И.А. Назарова, Л.В. Незамова - Донецк: ДонНТУ, 2016. - 100с.

Методические указания и задания к лабораторным работам по курсу "Компьютерной дискретной математики, часть 2" включают лабораторные работы по следующим основным темам курса:

- подграфы и изоморфизм;
- маршруты и связность в неориентированных графах;
- деревья и остовы неориентированных графов;
- циклы и обходы;
- ориентированные графы ;
- плоские и планарные графы;
- способы задания конечных автоматов;
- минимизация конечных автоматов .

Составители:

Назарова И.А., к.т.н., доцент

Незамова Л.В., асс.

Рецензент:

Лабораторная работа № 1

Подграфы и изоморфизм

Цель работы: изучение основных понятий теории графов и приобретение практических навыков определения изоморфизма и изоморфной вложимости графов, построение подграфов, независимых, доминирующих множеств и клик.

Теоретическая справка

Пусть V – некоторое непустое множество ($V \neq \emptyset$).

$V^{(2)}$ – множество всех его двухэлементных подмножеств, $\{u, v\}$ – неупорядоченная пара элементов множества V . $V^{(2)} = \{\{u, v\} | u, v \in V\}$.

Неориентированный граф G – пара множеств (V, E) , $E \subseteq V^{(2)}$, где

V – множество **вершин** графа G ,

E – множество **рёбер** графа G .

Если $|V|=p$, а $|E|=q$, то обозначают граф G , как (p, q) -граф или p -граф.

Смежные вершины графа G – вершины, соединенные ребром.

Смежные ребра графа G – ребра, имеющие общую вершину.

Инцидентные ребро и вершина – вершина является одним из концов ребра.

Конечный граф – множество вершин графа конечно.

Способы задания графов

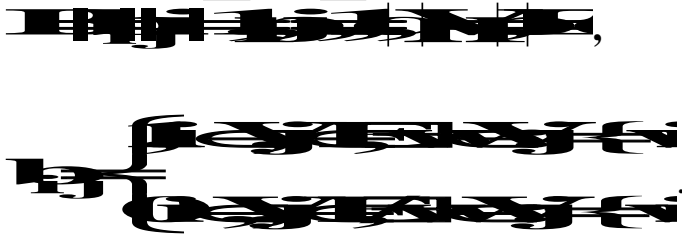
1. Явное перечисление множеств вершин V и ребер E .
2. Графический способ описания: прообраз вершины – точка, прообраз ребра – отрезок прямой или кривой.
3. Матричные способы описания.

1. Матрица смежности

~~Матрица смежности графа G – матрица $A = (a_{ij})$, где~~

~~$a_{ij} = 1$, если вершины v_i и v_j смежны, и $a_{ij} = 0$, в противном случае.~~

2. Матрица инцидентности

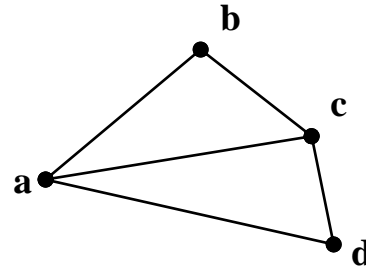


Например:

Задан граф $G=(V, E)$, где

$V=\{a, b, c, d\}$,

$E=\{ab, bc, ac, ad, dc\}$.



Матрица смежности A

	a	b	c	d
a	0	1	1	1
b	1	0	1	0
c	1	1	0	1
d	1	0	1	0

Матрица инцидентности B

	ab	bc	ac	ad	dc
a	1	0	1	1	0
b	1	1	0	0	0
c	0	1	1	0	1
d	0	0	0	1	1

Степени вершин графа

Степень вершины $\deg(v)$ графа G – число инцидентных ей ребер.

Максимальная степень всех вершин графа G – $\Delta(G)$:



Минимальная степень всех вершин графа G – $\delta(G)$:



Лемма о рукопожатиях

Сумма степеней всех вершин графа G четна и равна удвоенному числу ребер

Изолированная вершина графа G – вершина, степень которой равна 0.

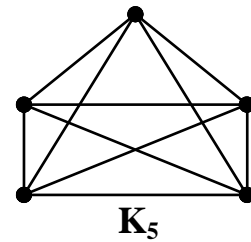
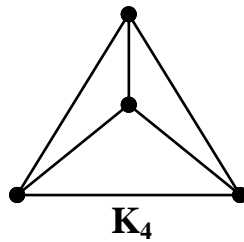
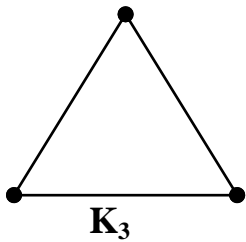
Висячая вершина графа G – вершина, степень которой равна 1.

Доминирующая вершина графа G – вершина, степень которой равна $p-1$, где p – количество вершин графа G .

Экстремальные графы

Полный граф – любые две вершины смежные или каждая вершина графа является доминирующей.

Обозначается, K_p .



Пустой граф – не имеет ребер. Обозначается через O_p .

Псевдограф – граф, содержащий петли и кратные ребра.

Мультиграф – граф, не содержащий петель, но с кратными ребрами.

Простой граф – конечный граф без петель и кратных ребер.

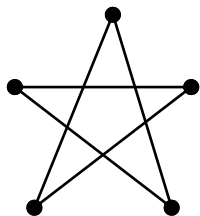
Далее, если особо не оговорено, рассматриваем только простые графы.

Нуль-граф – граф без вершин и без ребер.

Тривиальный граф – граф с одной вершиной, (1,0)-граф, K_1 , O_1 .

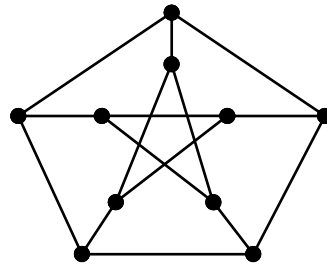
Однородный или регулярный граф – все вершины имеют равную степень.

deg =2



Граф Петерсона

deg=3

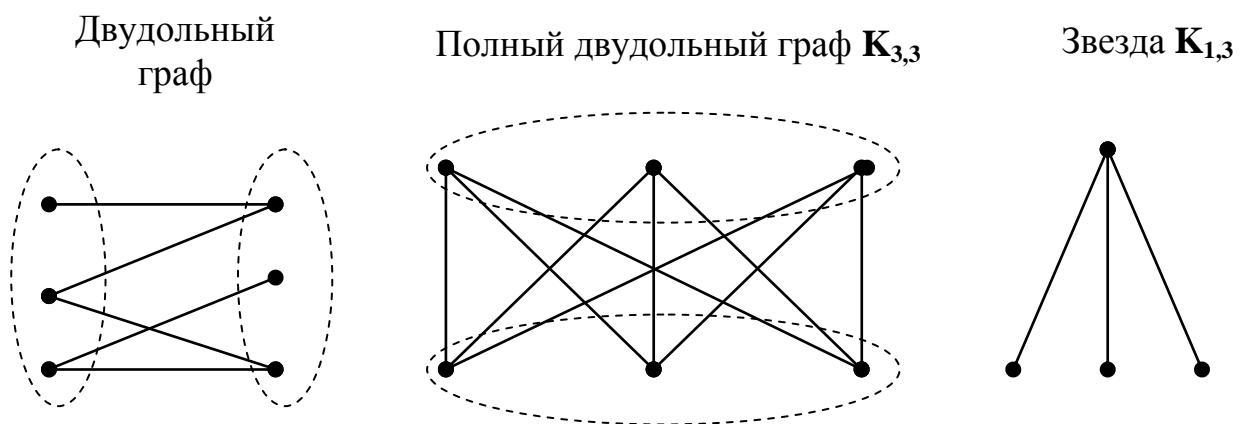


Например:

Двудольный граф (биграф) – множество вершин графа V можно разбить на два непересекающиеся подмножества V_1 и V_2 таких, что каждое ребро графа имеет одну концевую вершину в V_1 , а вторую – в V_2 , причем $V_1 \cap V_2 = \emptyset$, а $V_1 \cup V_2 = V$.

Полный двудольный граф – двудольный граф, у которого любые две вершины, входящие в разные доли V_1 и V_2 , смежны. Обозначается $K_{p,q}$, $p = |V_1|$, $q = |V_2|$.

Звезда – полный двудольный граф $K_{1,q}$.



Подграфы

Помеченный граф – граф, у которого каждой вершине поставлена в соответствие некоторая уникальная отметка (символ, цифра), иначе – **абстрактный**.

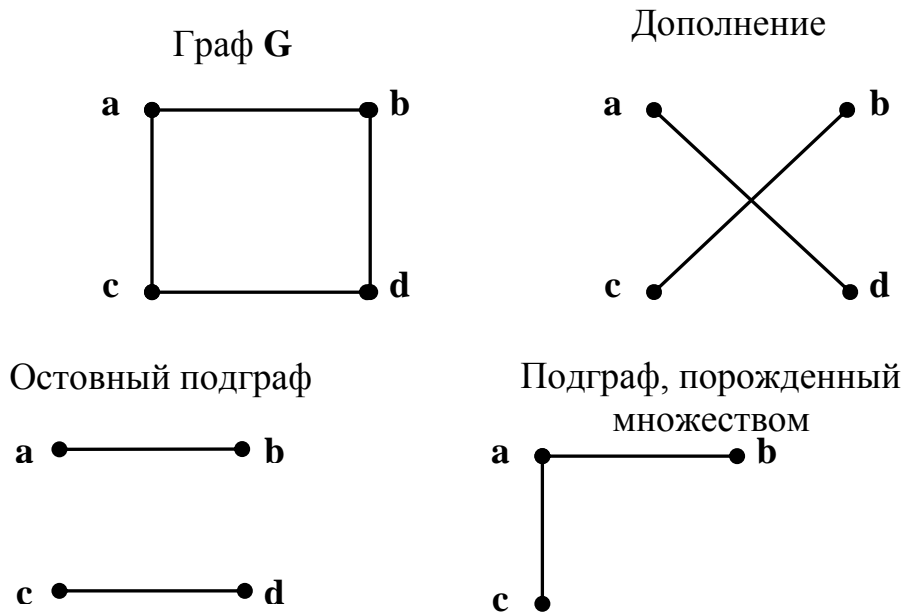
Дополнение графа G – граф $\bar{G} = (\bar{V}, \bar{E})$, такой, что $V = \bar{V}$, а $\bar{E} = V^{(2)} \setminus E$ (вершины смежные в \bar{G} несмежны в G и наоборот).

Подграф $G_1 = (V_1, E_1)$ графа $G = (V, E)$ – граф, у которого все вершины и ребра удовлетворяют следующим соотношениям $V_1 \subseteq V$, $E_1 \subseteq E$.

Остовный подграф графа G - подграф, содержащий все вершины графа G , множество ребер есть подмножество ребер графа G .

Порожденный подграф (порожденный подмножеством вершин V_1) – подграф, множество вершин которого $V_1 \subseteq V$, а множество ребер E_1 содержит все ребра графа G , инцидентные выбранным вершинам V_1 .

Например:

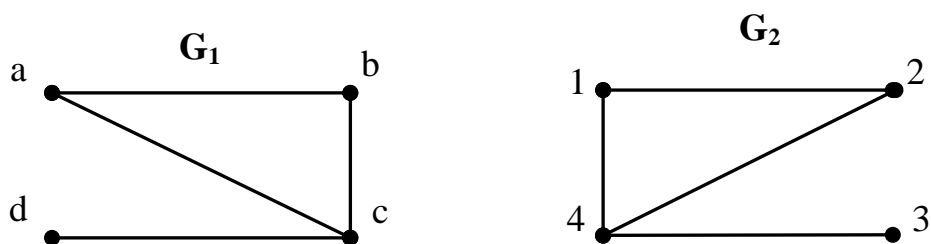


Изоморфизм графов

Изоморфные графы – существует взаимно однозначное соответствие между множествами их вершин или **биекция**, сохраняющая отношение смежности. Изоморфизм графов G и H : $G \cong H$.

Например:

Заданы два графа G_1, G_2 . Определить изоморфизм G_1, G_2 .



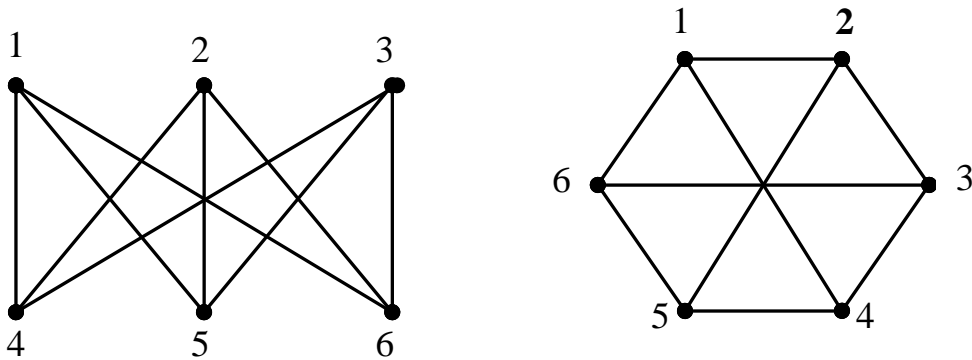
Решение:

Граф G_1 изоморфен графу G_2 , потому что существует биекция $\varphi: V_1 \rightarrow V_2$, сохраняющая отношение смежности.

Биекция φ :

$u \in V_1$	a	b	c	d
$\varphi(u) \in V_2$	2	1	4	3

Например: следующие графы изоморфны друг другу.

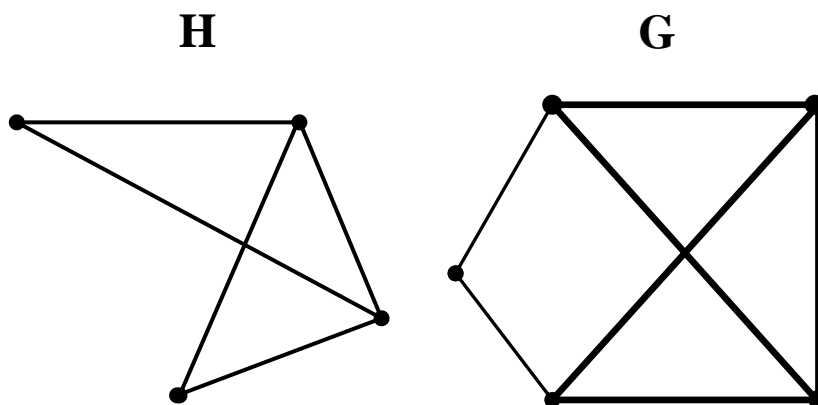


Биекция φ :

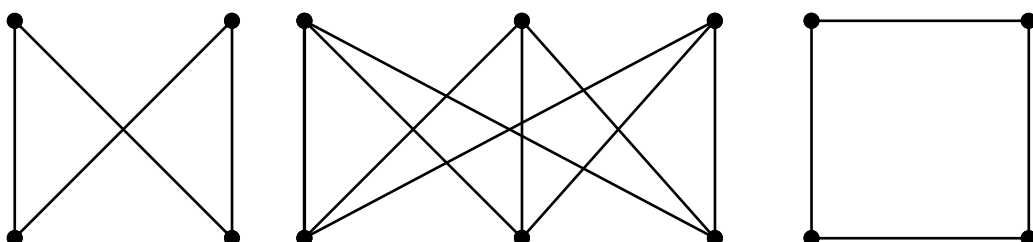
$u \in V_1$	1	2	3	4	5	6
$\varphi(u) \in V_2$	1	3	5	2	4	6

Граф G_1 **изоморфно вложим** в граф G , если граф G_1 является изоморфным некоторому порожденному подграфу графа G .

Например: граф G_1 изоморфно вложим в граф G .



Например: граф $K_{2,2}$ изоморфно вложим в $K_{3,3}$.



Независимые множества

Независимое множество вершин – подмножество вершин графа G : $S \subseteq V$ такое, что любые две вершины в нем несмежны, то есть никакая пара вершин не соединена ребром.

Подграф, порожденный независимым множеством – пустой граф.

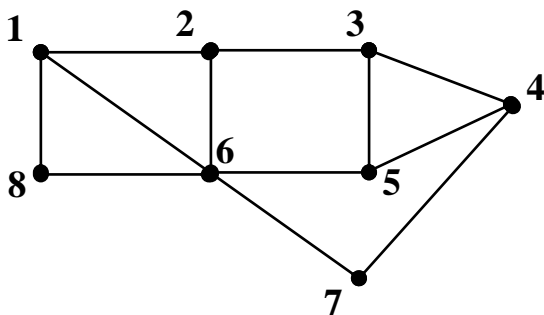
Максимальное независимое множество – не является собственным подмножеством другого независимого множества.

Наибольшее независимое множество – наибольшее по мощности среди всех независимых множеств.

Число независимости $\alpha(G)$ графа G – мощность наибольшего независимого множества.

Например:

Граф G :



Максимальные независимые множества

$$\begin{aligned}
 S_1 &= \{1, 3, 7\}; S_2 = \{1, 4\}; S_3 = \{1, 5, 7\}; \\
 S_4 &= \{2, 4, 8\}; S_5 = \{2, 5, 7, 8\}; \\
 S_6 &= \{3, 7, 8\}; S_7 = \{3, 6\}; \\
 S_8 &= \{4, 6\}.
 \end{aligned}$$

Наибольшее независимое множество графа G : $S_5 = \{2, 5, 7, 8\}$.

Число независимости графа G : $\alpha(G)=4$.

Клика

Клика – подмножество вершин графа G такое, что любая пара из этого множества является смежной.

Подграф, порожденный кликой – полный граф.

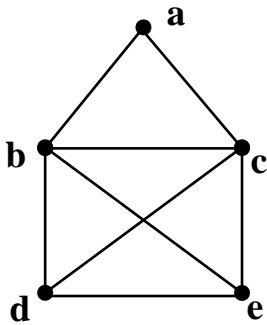
Максимальная клика – не является собственным подмножеством никакой другой клики графа G .

Наибольшая клика – наибольшая по мощности среди всех остальных клик графа G .

Кликовое число или плотность $\varphi(G)$ графа G – мощность наибольшей клики.

Например:

Граф G



Максимальные клики: $S_1=\{a,b,c\}$, $S_6=\{b,c,d,e\}$.

Наибольшая клика: $S_6=\{b,c,d,e\}$.

Кликовое число: $\varphi(G)=4$

Доминирующие множества

Доминирующее (внешне устойчивое) множество – подмножество $V' \subset V$ вершин графа такое, что каждая вершина из $V \setminus V'$ смежна с некоторой вершиной из V' .

Иначе, каждая вершина графа находится на расстоянии не более одного ребра от данного множества.

Минимальное доминирующее множество – нет другого доминирующего множества, содержащегося в данном.

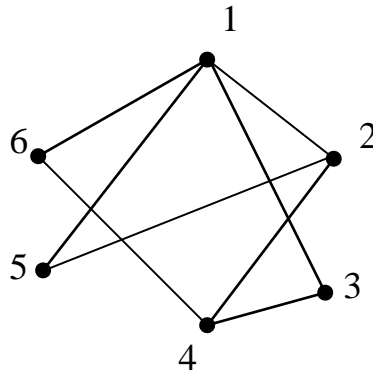
Наименьшее доминирующее множество – доминирующее множество с наименьшей мощностью.

Число доминирования $\beta(G)$ – мощность наименьшего доминирующего множества.

Например:

Примеры минимальных доминирующих множеств графа G :

$S_1 = \{1,4\}$; $S_2 = \{4,5\}$; $S_3 = \{2,3,6\}$; $S_4 = \{3,5,6\}$



Пример наименьшего доминирующего множества: $S_1 = \{1, 4\}$.

Число доминирования: $\beta(G) = 2$.

Задание к лабораторной работе

1. Используя алгоритм генерации варианта GV (приложение А), построить неориентированный граф $G: GV(7, \{2, 3\})$.

2. Описать граф матрицей смежности и матрицей инцидентности.

3. Изобразить графически граф G и его дополнение \bar{G} .

4. Построить произвольный остовный подграф и подграф, порожденный вершинами $\{1, 2, 5, 6, 7\}$;

5. Построить все помеченные 5-графы, изоморфно вложимые в граф G .

Определить классы изоморфных графов, построив биекцию их вершин.

Для каждого класса изоморфных графов привести рисунок абстрактного графа.

6. Построить все помеченные (5-7)-графы (до 20 штук), изоморфные некоторому подграфу G . Определить классы изоморфных графов, построив биекцию их вершин. Для каждого класса изоморфных графов привести рисунок абстрактного графа.

7. Найти все максимальные и наибольшие независимые множества исходного графа, определить число независимости.

8. Найти все максимальные и наибольшие клики данного графа. Определить плотность графа G .

9. Найти все минимальные и наименьшие доминирующие множества, определить число доминирования.

10. Найти полный двудольный подграф $K_{p,q}$, изоморфно вложимый в G с максимальным количеством вершин $p+q$ ($p \neq 1$). Найти звезду $K_{1,q}$, изоморфно вложимую в G с максимальным q .

Контрольные вопросы

1. Что такое неориентированный граф?
2. Определение подграфа, остовного и порожденного подграфа. Дополнение графа.
3. Изоморфизм графов.
4. Помеченные и абстрактные графы.
5. Максимальная и наибольшая клика. Кликовое число или плотность графа.
6. Максимальное и наибольшее независимое множество. Число независимости.
7. Полный, пустой, двудольный графы.
8. Число ребер в полном графе.
9. Число различных помеченных p -графов.
10. Число различных помеченных (p,q) -графов.

Лабораторная работа №2

Маршруты и связность в неориентированных графах

Цель работы: приобретение практических навыков в определении характеристик связности неориентированных графов, изучение различных алгоритмов нахождения кратчайших маршрутов.

Теоретическая справка**Маршруты в неориентированных графах**

Маршрут $M = (v_0, e_1, v_1, e_2, \dots, e_n, v_n)$ неориентированного графа $G=(V,E)$ – чередующаяся, конечная последовательность вершин и рёбер такая, что начинается и заканчивается вершиной и каждое ребро в маршруте соединяет две вершины маршрута – предыдущую и последующую.

Обозначения $M = (v_0, e_1, v_1, e_2, \dots, e_n, v_n)$ или $M = (v_0, v_1, \dots, v_n)$ или $M = (e_1, e_2, \dots, e_n)$, $M = (v_0 - v_n)$ -маршрут.

Концевые (терминальные) вершины маршрута – v_0 и v_n , **внутренние** – все остальные вершины.

Замкнутый маршрут $M = (v_0 - v_n)$ – концевые вершины совпадают ($v_0 = v_n$), иначе – **открытый** ($v_0 \neq v_n$).

Цепь – маршрут, все ребра которого различны (кроме, возможно, концевых).

Простая цепь – маршрут, все вершины которого, а, следовательно, и ребра, различны (кроме, возможно, концевых).

Цикл – замкнутая цепь, замкнутый маршрут без повторяющихся ребер.

Простой цикл – замкнутая простая цепь, $p \geq 3$, p – количество вершин.

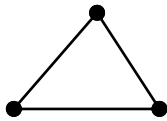
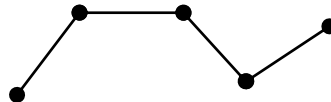
Утверждение 1.

Любой $(u-v)$ -маршрут неориентированном графе G содержит $(u-v)$ -простую цепь.

Утверждение 2.

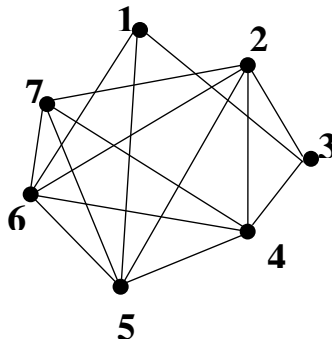
Любой цикл неориентированного графа содержит в себе простой цикл.

Граф, состоящий из одного простого цикла, обозначается C_p , граф, состоящий из одной простой цепи, обозначается P_p , здесь p – количество вершин графа.

 C_3  P_5 

Например:

Дан граф G . Привести примеры маршрута, цепи, простой цепи, замкнутого маршрута, цикла и простого цикла.



Маршрут $M1=(1,3,4,7,2,3,4,6)$.

Для маршрута $M1$ вершины 1 и 6 – концевые; 2,3,4,7 – внутренние.

Маршрут $M1$ – открытый маршрут, не цепь, не простая цепь (ребро (3,4) повторяется).

Маршрут $M2=(1,3,4,7,2,3,4,6,1)$.

Маршрут $M2$ – замкнутый маршрут, не цикл, не простой цикл (ребро (3,4) повторяется).

Маршрут $M3=(7,5,6,7,2,4)$.

Маршрут $M3$ – не простая цепь (вершина 7 повторяется).

Маршрут $M_4 = (7, 5, 6, 7, 2, 4, 7)$.

Маршрут M_4 – не простой цикл (вершина 7 повторяется).

Маршрут $M_5 = (1, 3, 4, 7, 6, 2)$.

Маршрут M_5 – простая цепь.

Маршрут $M_6 = (1, 3, 4, 5, 7, 6, 1)$.

Маршрут M_6 – простой цикл.

Связность в неориентированных графах

Связный неориентированный граф G – любая пара вершин соединена маршрутом (либо простой цепью) в G .

Компонента связности или **компонента** графа G – максимальный связный подграф графа G .

Две вершины v_i и v_j называются **связанными** в графе G , если в графе G существует маршрут между этими двумя вершинами. Вершина считается связанной сама с собой.

Связный граф состоит из одной компоненты связности.

Любой несвязный граф содержит, по крайней мере, две компоненты связности

Любой граф G является объединением своих компонент связности

Либо граф, либо его дополнение – связные графы

Число вершинной связности $\chi(G)$ – наименьшее число вершин, удаление которых приводит к несвязному или одновершинному графу.

Например: 

Число реберной связности $\lambda(G)$ – наименьшее число рёбер, удаление которых приводит к несвязному или одновершинному графу.

Например: 

Считается, что $\lambda(K_n) = n - 1$.

Точка сочленения (разделяющая вершина) – вершина v графа G , удаление которой увеличивает количество компонент связности графа G .

Мост – ребро, удаление которого приводит к увеличению числа компонент связности.

Неразделимый граф – граф без точек сочленения.

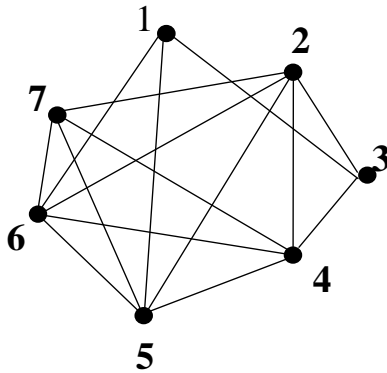
Блок – максимальный неразделимый подграф графа G .

Теорема Уитни

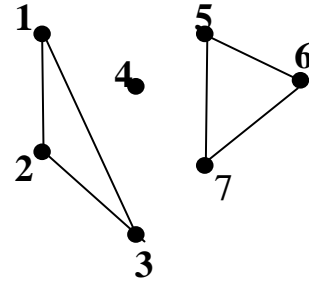
Для произвольного неориентированного графа справедливо неравенство: $\chi(G) \leq \lambda(G) \leq \delta(G)$,
где $\delta(G)$ – минимальная степень вершин графа G

Например:

Граф G :

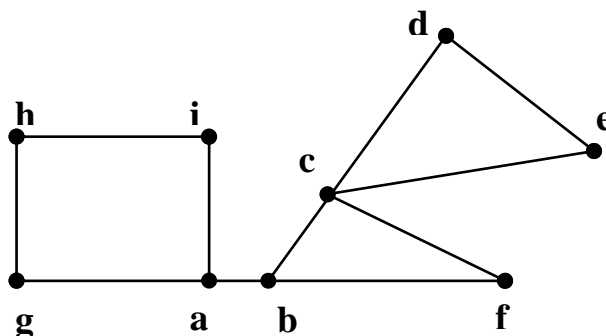


Граф R :



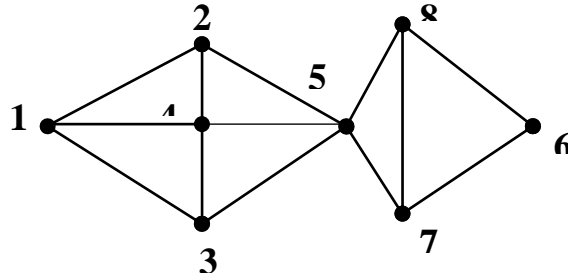
Граф G – связан. Граф R – несвязен, в графе R три компоненты связности, $R_1 = \{1, 2, 3\}$, $R_2 = \{4\}$, $R_3 = \{5, 6, 7\}$.

Граф G_1 :



Граф **G1**: точки сочленения – вершины **a, b, c**; мост – ребро **ab**. Граф **G1** не является неразделимым; блоки графа **G1**: **{c,d,e}**, **{c,b,f}**, **{a,g,h,i}**, **{a,b}**.

Граф **G2**:



Вершинная связность графа **G2** равна $\chi(G2)=1$ (удаление вершины **5** приводит к нарушению связности графа **G2**). Реберная связность графа **G2** равна $\lambda(G2)=2$ (удаление рёбер **(5,7)**, **(5,8)** приводит к нарушению связности графа **G2**).

Метрика в неорграфах

Длина маршрута – количество ребер, входящих в данный маршрут, каждое ребро учитывается столько раз, сколько раз оно входит в маршрут.

Обхват графа G – длина минимального простого цикла графа **G** (если он существует).

Окружение графа G – длина наибольшего простого цикла графа **G** (если он существует).

Расстояние $d(u,v)$ между двумя несовпадающими вершинами **u** и **v** – длина кратчайшей простой цепи, соединяющей эти вершины.

Геодезическая $\sigma(u,v)$ – кратчайшая простая цепь между вершинами **u** и **v**, доставляющая расстояние между этими вершинами.

Матрица расстояний

Матрица расстояний $D(G)$ – квадратная матрица $p \times p$, где **p** – количество вершин графа **G**: $D(G) = \|d_{ij}\|, i, j = \overline{1, p}, |V| = p, |E| = q$,

$$d_{ij} = \begin{cases} 0, & i = j \\ d(i, j), & i \neq j, \exists M = (i - j) \\ \infty, & i \neq j, \nexists M = (i - j) \end{cases}$$

Эксцентриситет $e(v)$ вершины v графа G – длина максимальной геодезической, исходящей из вершины v :

$$e(v) = \text{MAX}_{u \in V} d(v, u).$$

Диаметр $D(G)$ графа G – максимальный среди всех эксцентриситетов вершин графа G :

$$D(G) = \text{MAX}_{v \in V} e(v).$$

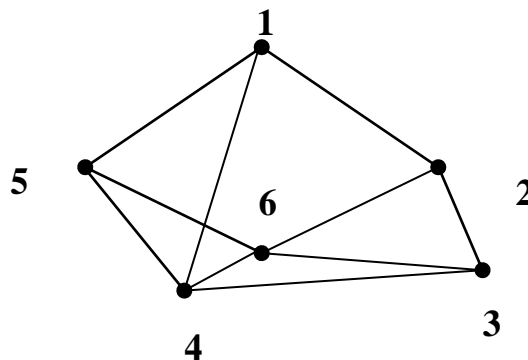
Радиус $R(G)$ графа G – минимальный среди всех эксцентриситетов вершин графа G :

$$R(G) = \text{MIN}_{v \in V} e(v).$$

Периферия графа G – множество вершин графа G , у которых эксцентриситет равен диаметру.

Центр графа G – множество всех вершин графа G , у которых эксцентриситет равен радиусу.

Например: Граф G : вес каждого ребра равен 1.



Матрица расстояний D_G

$i \backslash j$	1	2	3	4	5	6	$e(j)$
1	0	1	2	1	1	2	2
2	1	0	1	2	2	1	2
3	2	1	0	1	2	1	2
4	1	2	1	0	1	1	2
5	1	2	2	1	0	1	2
6	2	1	1	1	1	0	2

Диаметр G : $D(G)=2$. Радиус G : $R(G)=2$.

Периферия графа $G = \{1, 2, 3, 4, 5, 6\}$.

Центр графа $G = \{1,2,3,4,5,6\}$.

Обхват графа $G = 3$.

Окружение графа $G = 6$, максимальный простой цикл, который содержит все вершины графа $G (1,2,3,4,6,5,1)$.

Задание к лабораторной работе

Исходные данные: граф $G1: GV(13,\{6,7\})$

Алгоритм генерации варианта $GV(p,X)$ описан в приложении А.

1. Определить, является ли граф $G1$ связным.
2. Для максимальной компоненты графа $G1$:
 - а) выделить маршрут не цепь, замкнутый маршрут не цепь, цепь, простую цепь, цикл, простой цикл;
 - б) определить обхват и окружение;
 - с) найти вершинную и реберную связность.
3. Для каждой компоненты графа $G1$:
 - а) построить матрицу расстояний, определить эксцентриситеты вершин, радиус, диаметр, центр, периферию, диаметральную цепь;
 - б) определить, является ли она неразделимой, выделить блоки, найти точки сочленения и мосты.

Контрольные вопросы

1. Привести пример графа, удовлетворяющего строгому неравенству теоремы Уитни.
2. Привести примеры графов, которые имеют все периферийные и все центральные вершины.
3. Что такое эксцентриситет?
4. Чем диаметр графа отличается от его радиуса (дайте их определения)?
5. Чем простая цепь отличается от цикла?
6. Что такое маршрут?
7. Что такое число реберной связности?
8. Дайте определения моста и цикла.

Лабораторная работа № 3

*Поиск кратчайших маршрутов***Алгоритм нахождения кратчайших маршрутов**

Задача о кратчайшем пути состоит в нахождении кратчайшего пути от заданной начальной вершины (s) до заданной конечной вершины (t), при условии, что такой путь существует.

Задачи данного типа имеют следующие модификации:

- для заданной начальной вершины s найти кратчайшие пути от s ко всем другим вершинам графа;
- найти кратчайшие пути между всеми парами вершин.

Для решения этих задач рассмотрим граф $G = (V, E)$, $|V| = p$, $|E| = q$, ребра которого имеют определенные веса (стоимости).

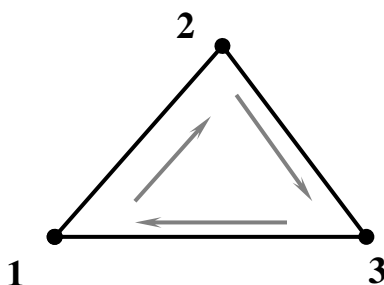
Веса ребер задаются матрицей $C = \|c_{ij}\|$, $i, j = \overline{1, p}$. Элементы c_{ij} матрицы весов C могут быть положительными, отрицательными или нулями, поэтому для большинства алгоритмов поиска кратчайших маршрутов существует ограничение: в G не должно быть циклов с суммарным отрицательным весом (в этом случае кратчайшего маршрута не существует).

Например, если веса ребер графа G_1 составляют соответственно $\overline{1, 2, 3, 1, 3}$, то имеется цикл суммарного отрицательного веса $\overline{1, 2, 3, 1, 3}$ и задача не имеет решения.

Введем обозначения, пусть $\Gamma(v_i)$ – множество вершин графа G , которые смежны с вершиной v_i .

Так, для графа G_1 имеем $\Gamma(1) = \{2, 3\}$.

G_1 :



Алгоритм Дейкстры ($c_{ij} \geq 0$)

В общем случае этот метод основан на том, что вершинам приписываются временные пометки. Пометки обозначают длины путей от начальной вершины s к данной вершине (причем временные пометки являются верхними границами длин путей).

Величины этих пометок постепенно уменьшаются с помощью итерационной процедуры (т.е. процедуры, в которой постоянно повторяется некоторый набор операций).

На каждом шаге итерации одна из временных пометок становится постоянной, т.е. такой, которая обозначает точную длину кратчайшего пути от s к рассматриваемой вершине.

Рассмотрим этот алгоритм для случая, когда вес каждого ребра графа неотрицателен ($\forall i, j \in \Gamma \Rightarrow c_{ij} \geq 0$).

Алгоритм Дейкстры

Пусть $l(i)$ – пометка вершины i .

Шаг 1. Положить $l(s) = 0$ и считать эту пометку постоянной.

Положить $l(i) = \infty$ для всех $i \neq s$ и считать эти пометки временными. Положить $p = s$.

Шаг 2. Для всех $i \in \Gamma(p)$, пометки которых временные, заменить пометки в соответствии со следующим выражением:

$$l(i) = \min(l(i), l(p) + c_{pi})$$

Шаг 3. Среди всех вершин со временными пометками найти такую, для которой: $l(i^*) = \min_{i \in \Gamma(p)} l(i)$.

Шаг 4. Считать пометку вершины i^* постоянной и положить $p = i^*$.

Шаг 5.

Шаг 5. 1. Необходимо найти кратчайший маршрут только от s к t .

При $p = t$ алгоритм заканчивает работу, $l(p)$ является длиной кратчайшего маршрута от s к t . При $p \neq t$ перейти к шагу 2.

Шаг 5.2. Необходимо найти кратчайшие маршруты от s ко всем остальным вершинам графа.

Если все вершины получили постоянные пометки, то алгоритм заканчивает работу.

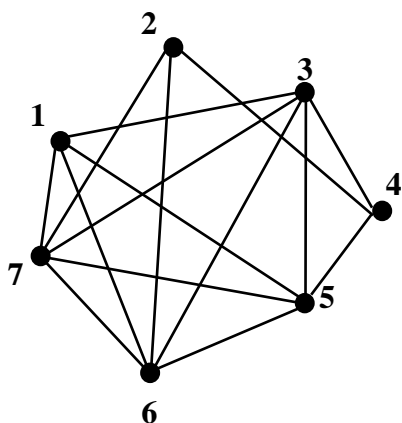
Если некоторые вершины имеют временные пометки, то перейти к шагу 2.

Как только длины кратчайших путей будут найдены, сами пути можно получить при помощи рекурсивной процедуры с использованием соотношения: $l(i') + c(i', i) = l(i)$. Поскольку вершина i' непосредственно предшествует вершине i в кратчайшем пути от s к i , то для любой вершины i соответствующую вершину i' можно найти как одну из оставшихся вершин, для которой выполняется: $l(i') + c(i', i) = l(i)$.

Например: граф G задан графически и матрицей весов.

Найти кратчайшие расстояния в графе от первой вершины ко всем остальным.

Замечание: постоянные пометки будем выделять знаком +.



Матрица весов для графа C_G :

	1	2	3	4	5	6	7
1	0	0	4	0	15	13	1
2	0	0	0	4	0	5	7
3	4	0	0	9	12	13	2
4	0	4	9	0	3	0	0
5	15	0	12	3	0	2	11
6	13	5	13	0	2	0	12
7	1	7	2	0	11	12	0


Шаг 1. $l(1)=0$

Устанавливаем нулевую пометку для вершины **1**, считаем ее постоянной.

$l(2)=l(3)=\dots=l(7)=\infty$

Устанавливаем временные пометки для вершин (2,...,7).

$p=1$

Шаг 2.  – все вершины, смежные вершине 1 имеют временные пометки

$$l(3)=\min(\infty,0+4)=4$$

Изменяем временные

$$l(5)=\min(\infty,0+15)=15$$

пометки в соответствии с

$$l(6)=\min(\infty,0+13)=13$$

выражением:

$$l(7)=\min(\infty,0+1)=1$$



Шаг 3.


 , соответствует вершине $i^* = 7$.

Шаг 4.

$l(7)=1^+$ – вершина 7 получает постоянную пометку

$p=7$

Шаг 5. Не все вершины имеют постоянные пометки, а только {1,7}, алгоритм продолжает работу, переходим к шагу 2.

Шаг 2.  среди них только вершина 1 имеет постоянную пометку.

$$l(2)=\min(\infty, 1^++7)=8$$

$$l(3)=\min(4, 1^++2)=3$$

$$l(5)=\min(15, 1^++11)=12$$

$$l(6)=\min(13, 1^++12)=13$$

Шаг 3.

$l(i^*) = \min(8, 3, \infty, 12, 13) = 3$, соответствует вершине $i^* = 3$.

Шаг 4.

$l(3)=3^+$ – вершина 3 получает постоянную пометку.

$p=3$

Шаг 5. Не все вершины имеют постоянные пометки, а только {1,3,7}, алгоритм продолжает работу, переходим к шагу 2.

Шаг 2. ~~10-13-9-5~~. Постоянные пометки имеют вершины {1,3,7}, их исключаем из рассмотрения.

$$l(4) = \min(\infty, 3^+ + 9) = 11$$

$$l(5) = \min(12, 3^+ + 12) = 12$$

$$l(6) = \min(13, 3^+ + 13) = 13$$

Шаг 3. Среди всех вершин с временными пометками, а это множество {2,4,5}, находим:

$$l(i^*) = \min(8, 11, 12, 13) = 8, \text{ вершина } 2 \text{ получает постоянную пометку.}$$

Шаг 4.

$$l(2) = 8^+ \text{ – вершина } 2 \text{ получает постоянную пометку.}$$

$$p = 2$$

Шаг 5. Не все вершины имеют постоянные пометки, а только {1,2,3}, алгоритм продолжает работу, переходим к шагу 2.

Шаг 2. ~~10-13-9-5~~, вершина 7 имеет постоянную пометку, поэтому исключается из рассмотрения.

$$l(4) = \min(12, 8^+ + 4) = 12$$

$$l(6) = \min(13, 8^+ + 5) = 13$$

$$l(4) = l(5) = 12^+$$

$$p = 4, 5$$

Можно выбирать как вершину 4, так и вершину 5. Выбираем вершину 4.

$$l(4) = 12^+$$

$$p = 4$$

Множество вершин с постоянными пометками равно {1,2,3}.

Шаг 2. ~~10-13-9-5~~, вершины 2 и 3 имеют постоянную пометку, следовательно, не рассматриваются.

$$l(5) = \min(12, 12^+ + 3) = 12$$

$$l(i^*) = \min(12, 13) = 12, \text{ вершина } 5 \text{ получает постоянную пометку.}$$

$$l(5) = 12^+$$

$$p = 5$$

Множество вершин с постоянными пометками равно {1,2,3,5}.

Шаг 2. ~~10 15 13 5~~, все вершины, кроме **6** имеют постоянные пометки.

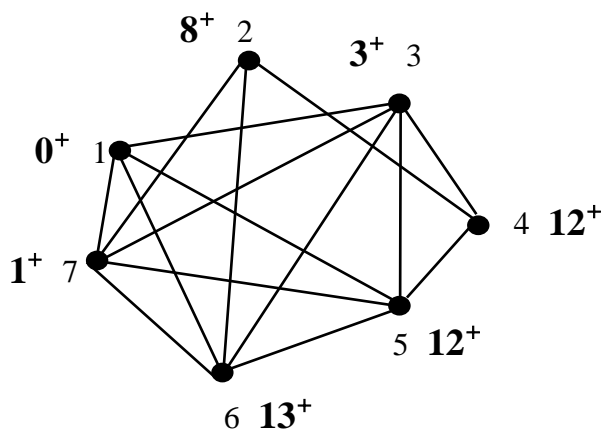
$$l(6) = \min(13, 12^+ + 2) = 13$$

$$\underline{l(6) = 13^+}$$

$$p = 6$$

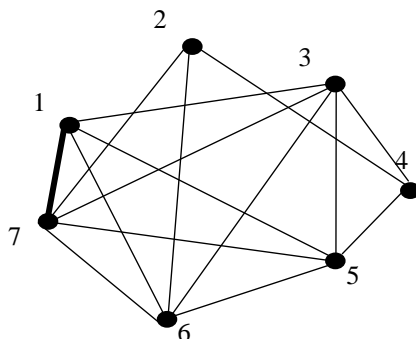
Множество вершин с постоянными пометками равно **{1, 2, 3, 4, 5, 6}**.

Шаг 5. Все вершины имеют постоянные пометки, алгоритм закончил работу. Найдены кратчайшие расстояния от вершины **1** ко всем остальным вершинам в графе.

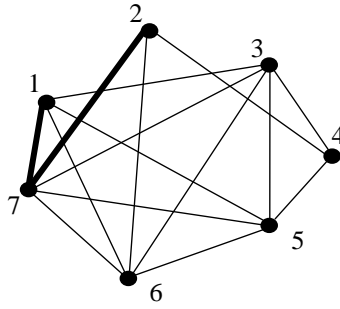


Таким образом, кратчайшие маршруты от вершины 1:

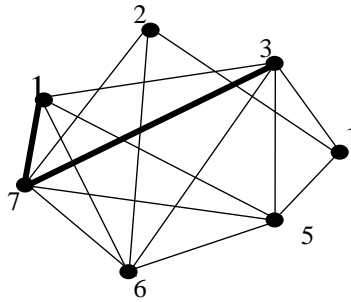
$$1-7 = (1, 7) = 1;$$



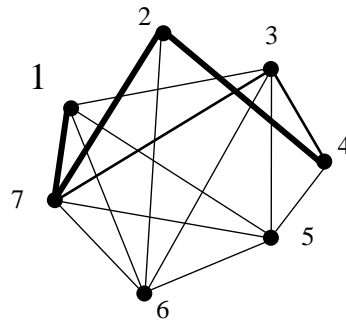
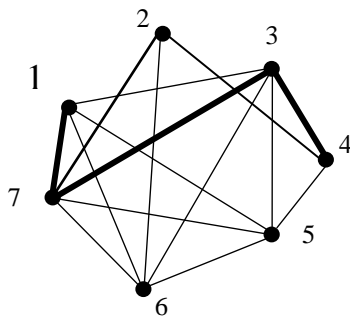
$$1-2 = (1, 7, 2) = 8;$$



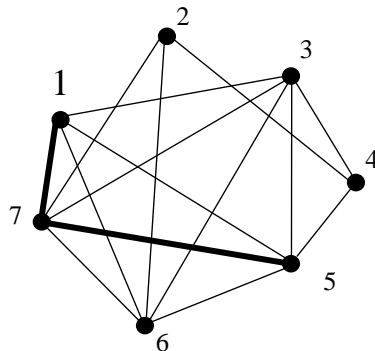
$1-3=(1,7,3)=3;$



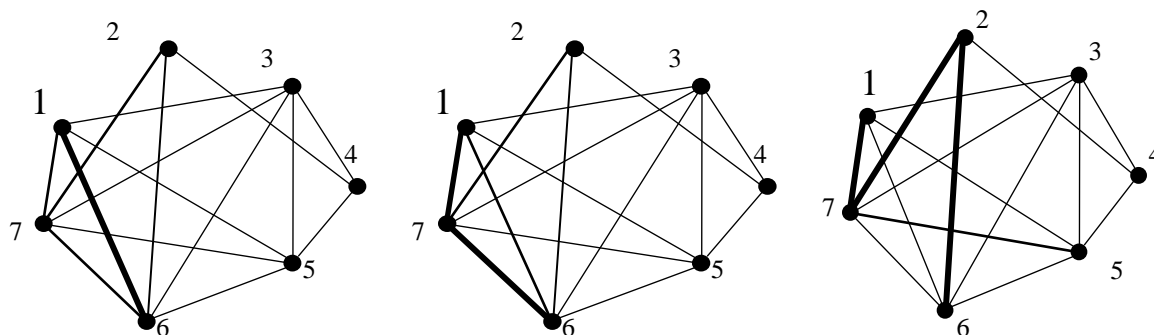
$1-4=(1,7,3,4)=(1,7,2,4)=12;$



$1-5=(1,7,5)=12;$



$1-6=(1,6)=(1,7,6)=(1,7,2,6)=13;$



Задание к лабораторной работе

Исходные данные: граф $G_2: GV(7, \{2,3\})$.

Ребра графа G_2 взвешены соответствующими элементами матрицы Y .

В графе G_2

- а) построить кратчайшие маршруты от произвольной вершины ко всем остальным при помощи алгоритма Дейкстры;
- б) построить кратчайшие маршруты от произвольной вершины ко всем остальным при помощи алгоритма Форда;
- в) построить кратчайшие маршруты при помощи алгоритма Флойда. При построении вести две матрицы – матрицу маршрутов и матрицу расстояний.

Контрольные вопросы

1. Назначение алгоритма Дейкстры, Форда и Флойда.
2. Чем временные пометки отличаются от постоянных пометок?
3. Какие ограничения применяют к алгоритмам?
4. Сформулируйте по шагам алгоритм Дейкстры, Форда и Флойда.

Лабораторная работа №4

Деревья и остовы неориентированных графов

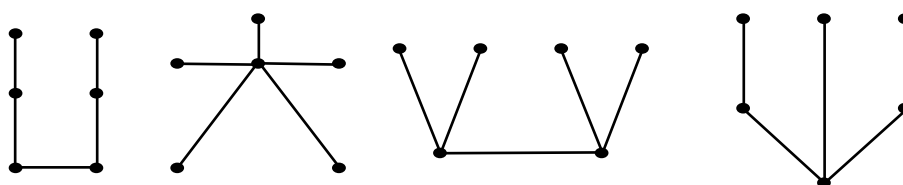
Цель работы: приобретение практических навыков в работе с деревьями неориентированных графов, изучение различных алгоритмов нахождения остовов кратчайших маршрутов.

Теоретическая справка

Дерево – связный граф, не содержащий циклов.

Лес (ациклический граф) – произвольный граф, не содержащий циклов.

Компонентами леса являются деревья.

**Теорема о дереве (5 различных определений дерева)**

Для любого (p, q) -графа G следует, что утверждения (1-5) эквивалентны:

1. Граф G – дерево, связный ациклический граф.
2. Любые две несовпадающие вершины графа G соединены единственной простой цепью.
3. Граф G – связен, и количество ребер в нем на 1 меньше, чем количество вершин: $q = p - 1, p = q + 1$.
4. Граф G – ациклический, и $q = p - 1, p = q + 1$.
5. Граф G – ациклический, и если любую пару его несовпадающих несмежных вершин соединить ребром e , то полученный граф $G + e$ будет содержать ровно один цикл.

Теорема о висячих вершинах дерева

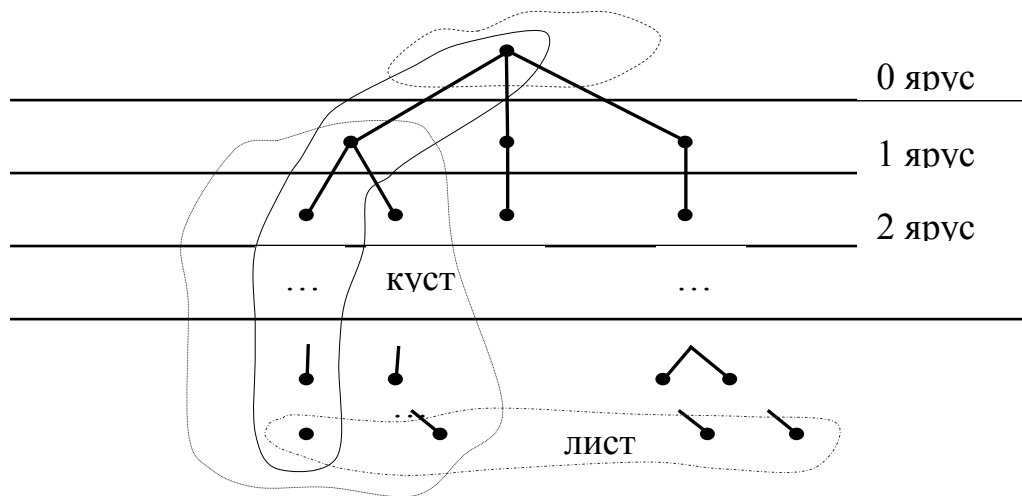
В любом нетривиальном ($p \geq 2$) дереве имеется, по крайней мере, 2 висячие вершины.

Теорема о центре дерева.

Центр любого дерева состоит либо из 1, либо из 2 смежных вершин.

Ярусная форма представления деревьев

Пусть существует вершина v_0 – корень графа $G = (V, E)$, $v_0 \in V$. Эту вершину ориентируем. На i -й ярус помещаются вершины с расстоянием от корня, равным i . Концевые висячие вершины будем называть **листами**, а геодезические от корня к листу называют **кустом**.

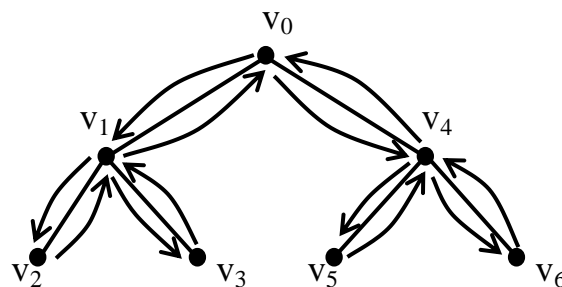


I

Способы обхода деревьев

1. Способ обхода в глубину

Поиск в глубину подразумевает просмотр ветвей дерева.



Начинаем поиск с некоторой фиксированной вершины v_0 .

Находим вершину u смежную с v_0 и повторяем весь процесс, начиная с вершины u :

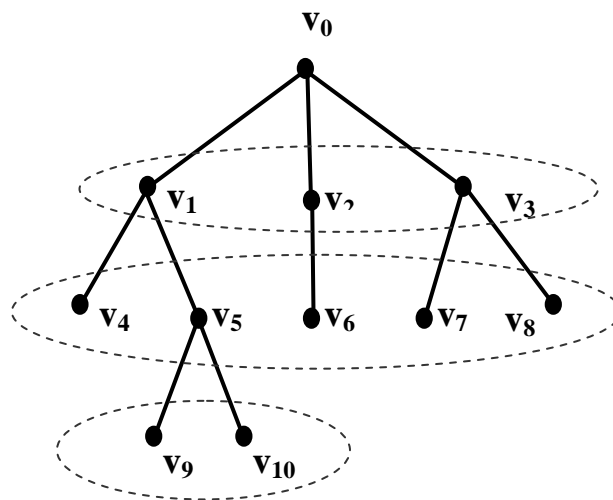
- если существует не просмотренная вершина, смежная с вершиной u , рассматриваем ее и, начиная с нее, выполняем поиск;

- если не существует ни одной новой вершины, смежной с u , то говорят, что вершина u использована, и делается возврат в вершину, из которой мы попали в вершину u и продолжаем процесс;

- если на каком-то шаге $u = v_0$, и нет ни одной использованной вершины, то алгоритм заканчивает работу.

2. Способ обхода в ширину

Подразумевает просмотр вершин по ярусам в иерархическом представлении. Здесь под использованием вершины подразумевается просмотр всех «соседей» данной вершины.



Остовы

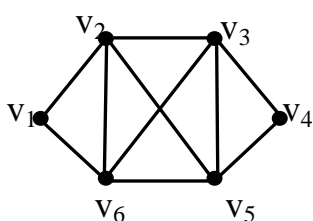
(наличие деревьев в произвольном графе)

Остов (каркас, скелет) графа G – это остовный подграф графа G , задающий дерево на каждой компоненте связности графа G .

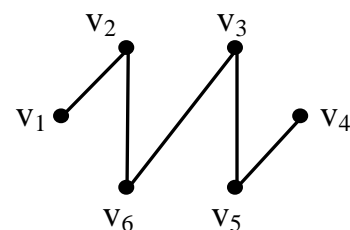
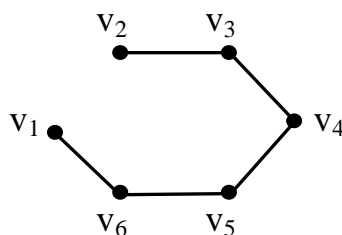
Для связного графа **остов** – это дерево, покрывающее все вершины исходного графа.

Пусть есть некоторый граф G :

G :



ОСТОВЫ:



Ребра остова T некоторого графа G называются **ветвями**, а ребра графа G , не вошедшие в ост, называются **хордами**.

Матричная теорема Кирхгофа

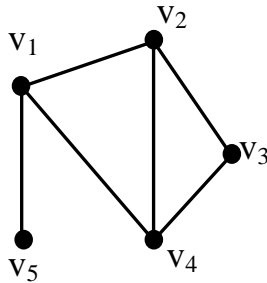
Матрица Кирхгофа – это квадратная матрица

$$M = \|m_{ij}\|, \quad i, j = 1, \dots, p, \quad m_{ij} = \begin{cases} \deg v_i, & i = j \\ -a_{ij}, & i \neq j \end{cases}$$

где a_{ij} – соответствующий элемент матрицы смежности.

Для графа G матрица Кирхгофа такова:

G :



$M =$

	v_1	v_2	v_3	v_4	v_5
v_1	3	-1	0	-1	-1
v_2	-1	3	-1	-1	0
v_3	0	-1	2	-1	0
v_4	-1	-1	-1	3	0
v_5	-1	0	0	0	1

Сумма элементов любой строки и столбца равна 0.

Теорема Кирхгофа

Число остовных деревьев в связном графе G порядка p , $p \geq 2$ равно алгебраическому дополнению любого элемента матрицы Кирхгофа.

В связном помеченном графе все алгебраические дополнения матрицы Кирхгофа равны между собой и определяют общее число помеченных остовов этого графа.

Следствие из теоремы Кирхгофа

При $p > 1$ p – количество вершин) число остовов полного графа K_p равно p^{p-2} .

Теорема А. Кэли (1897 г.)

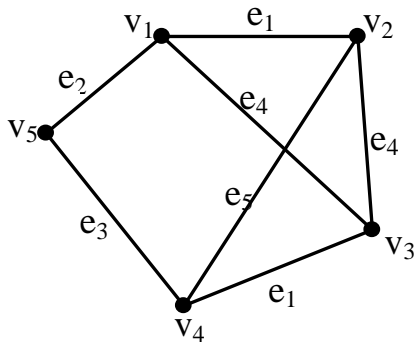
Число помеченных деревьев порядка p равно p^{p-2} .

Алгоритмы поиска остовов кратчайших маршрутов

Имеется граф, заданный матрицей весов, т.е. существует $G = (V, E)$, $C = \|c_{ij}\|, i, j = \overline{1, p}$, где p – количество вершин графа, c_{ij} – вес ребра $\{i, j\}$, если оно существует. Требуется найти остов с минимальным суммарным весом ребер.

Алгоритм Краскала

1. Упорядочиваем ребра графа $G = (V, E)$ в порядке неубывания их весов.
2. Строим пустой граф O_p , где p – количество вершин исходного графа
3. На каждом шаге к уже сформированному текущему графу, добавляется ребро из списка ребер исходного графа с минимальным весом. Добавляемое ребро не должно приводить к образованию цикла.
4. Алгоритм заканчивает работу, если количество ребер в формируемом графе станет равным $p - 1$.



Пример: алгоритм Краскала

Список ребер

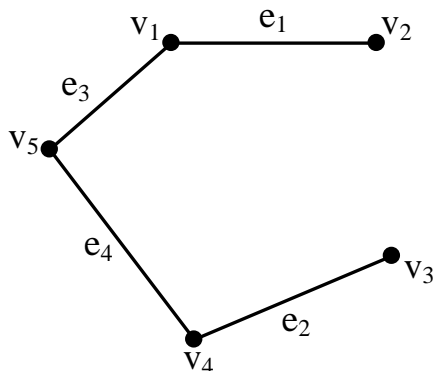
$$e_1 = (1, 2), (4, 3) - 1$$

$$e_2 = (1, 5) - 2$$

$$e_3 = (4, 5) - 3$$

$$e_4 = (1, 3), (2, 3) - 4$$

$$e_5 = (2, 4) - 5$$



Остов кратчайших маршрутов:

$$M = ((1, 2), (4, 3), (1, 5), (4, 5))$$

$$\text{Общий суммарный вес } \sum = 7.$$

Алгоритм Прима

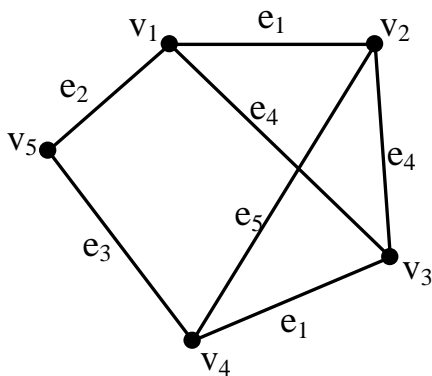
Отличается от алгоритма Краскала тем, что на каждом шаге строится не просто граф без циклов, но дерево.

1. Упорядочиваем ребра графа G в порядке неубывания их весов.
2. Строим пустой граф O_p , где p – количество вершин исходного графа

3. На каждом шаге к текущему, уже сформированному графу, добавляем ребро исходного графа с минимальным весом. Ребро не должно образовывать цикл и нарушать связность. Для этого список ребер каждый раз просматривается, начиная с 1 ребра.

4. Алгоритм заканчивает работу, если количество ребер в формируемом графе станет равным $p - 1$.

Пример: алгоритм Прима



Список ребер

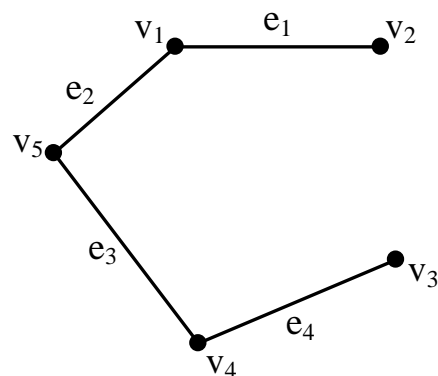
$$e_1=(1, 2), (4, 3) - 1$$

$$e_2=(1, 5) - 2$$

$$e_3=(4, 5) - 3$$

$$e_4=(1, 3), (2, 3) - 4$$

$$e_5=(2, 4) - 5$$



Остов кратчайших маршрутов: $M=((1, 2), (1, 5), (5, 4), (4, 3))$

Общий суммарный вес $\sum = 7$.

Задание к лабораторной работе

Исходные данные: граф $G1 = GV(5, \{2, 3\})$,

граф $G2 = GV(13, \{6, 7\})$.

1. Для графа $G1$ составить матрицу Кирхгофа и посчитать количество помеченных остовов.
2. Для графа $G2$ построить дерево обхода вершин графа (использовать алгоритм в ширину, в глубину).
3. Для графа $G2$ решить задачу построения остовов кратчайших маршрутов, используя алгоритмы Прима и Краскала. В качестве весов ребер использовать элементы вспомогательной матрицы Y .
4. Сгенерировать все различные абстрактные, не изоморфные друг другу деревья порядка (4-7).
5. Разделить множество деревьев на 2 подмножества с одной и с двумя центральными вершинами.

Алгоритм генерации варианта $GV(p, X)$ описан в приложении А.

Контрольные вопросы

1. Привести определение дерева и леса.
2. Способы обхода деревьев.
3. Какие вершины дерева называются центром?
4. Что называется остовом?
5. Как можно определить число остовных деревьев?
6. Чем отличаются алгоритмы Краскала и Прима?

Лабораторная работа №5

Циклы и обходы

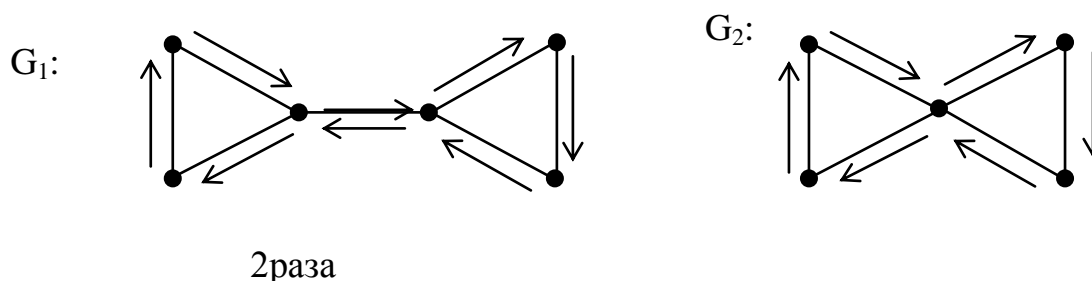
Цель работы: приобретение практических навыков в нахождении эйлеровых и гамильтоновых циклов в неориентированных графах, решение задач «китайского почтальона» и коммивояжера

Теоретическая справка**Обходы графа****Эйлеровы циклы**

Эйлеров цикл – цикл, содержащий все рёбра исходного графа (каждое ребро используется ровно 1 раз).

Эйлеров граф – связный граф, содержащий эйлеров цикл.

Например:



**Теорема Эйлера или критерий существования в графе
эйлерового цикла**

**Связный граф G является эйлеровым тогда и только тогда,
когда степени всех его вершин четны**

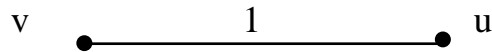
Следствие №1. Для связного графа G множество ребер можно разбить на простые циклы, если граф эйлеров.

Следствие №2. Для того чтобы связный граф G покрывался единственной эйлеровой цепью, необходимо и достаточно, чтобы он содержал ровно 2

вершины с нечетной степенью. Тогда цепь начинается в одной из этих вершин и заканчивается в другой.

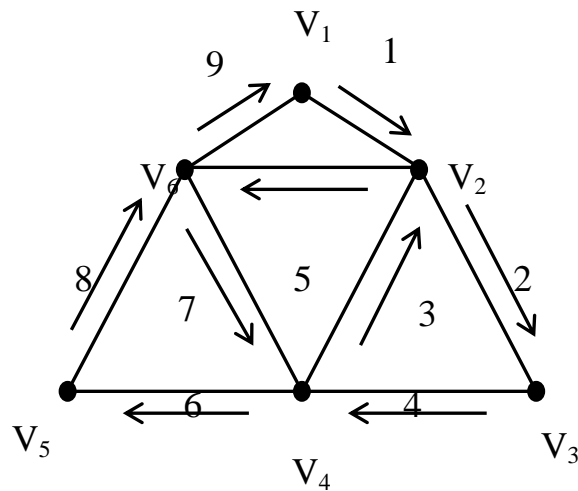
Алгоритм построения эйлерового цикла или алгоритм Флёрри

1. Начиная с любой вершины v , присваиваем ребру vu №1. Вычеркиваем это ребро из списка ребер и переходим к вершине u .



2. Пусть w – вершина, в которую мы пришли в результате выполнения шага 1 и k – номер, присвоенный очередному ребру на этом шаге. Выбираем произвольное ребро инцидентное вершине w , причем мост выбираем только в крайнем случае, если других возможностей выбора ребра не существует. Присваиваем ребру номер $k+1$ и вычеркиваем его. Процесс длится до тех пор, пока все ребра не вычеркнуты.

Например:



Эйлерова цепь – цепь с концевыми вершинами (a, b) , начинающаяся в вершине a , заканчивающаяся в вершине b и содержащая каждое ребро исходного графа ровно 1 раз.

Гамильтоновы циклы

Гамильтонов цикл – простой цикл, содержащий каждую вершину графа.

Гамильтонов граф – граф, содержащий гамильтонов цикл.

Достаточные условия существования гамильтонова цикла в графе

Теорема Дирака

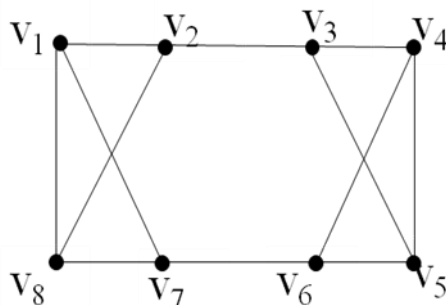
Если число вершин графа $p \geq 3$ и для любой вершины выполняется условие

$$\forall v_i, i = \overline{1, p}; \deg v_i \geq \frac{p}{2}, \text{ то граф } G \text{ – гамильтонов}$$

Теорема Оре

Если число вершин графа $p \geq 3$ и для любых двух несмежных вершин u и v выполняется неравенство: $\deg u + \deg v \geq p$, то граф G – гамильтонов

Например:



$$p=8, \deg v_i = 3, 3 \not\geq \frac{8}{2} = 4.$$

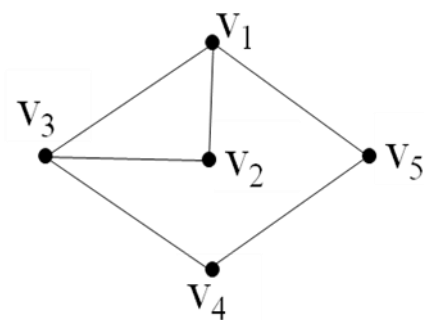
В данном графе не выполняется условие теоремы Дирака, но существует гамильтонов цикл: $M = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_1)$.

Дерево полного перебора

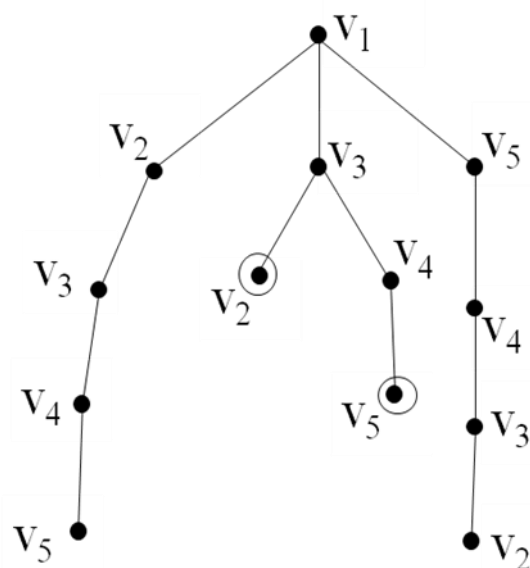
Дерево полного перебора строится для определения гамильтоновых циклов в графе и используется при небольшом количестве вершин данного графа.

Построение дерева производится слева направо:

Граф G:



Дерево полного перебора



Гамильтоновы циклы графа G:

1. $(v_1, v_2, v_3, v_4, v_5, v_1)$;
2. $(v_1, v_5, v_4, v_3, v_2, v_1)$

Алгоритм перебора Робертса и Флореса

(поиск гамильтонова цикла)

Задача: задан граф G, найти все гамильтоновы циклы исходного графа.

Идея алгоритма.

– выбирается некоторая начальная вершина графа. Пусть исходной будет v_1 .

Эта вершина образует первый элемент множества S; $S = \{v_1\}$.

– множество S на каждом шаге будет хранить уже найденные вершины гамильтоновой цепи. К S добавляется первая вершина в столбце, соответствующем v_1 . Пусть эта вершина a.

– затем к S добавляется первая “возможная” вершина в столбце a . Пусть это вершина b . $S = \{v_1, a, b\}$.

Под “возможной” понимается вершина, которой нет в S .

Существует **2 принципа**, препятствующих включению некоторой вершины во множество S .

Пусть множество S имеет вид: $S = \{v_1, a, b, c \dots v_{r-1}, v_r\}$

1. Если в столбце v_r нет возможных вершин (множество S нельзя расширить).
2. Цепь, определенная последовательностью вершин S имеет длину $p - 1$, где p – количество вершин графа, т. е. она является гамильтоновой цепью.

В случае 2 тоже 2 варианта.

а) В графе G существует ребро (v_r, v_1) , следовательно, найден гамильтонов цикл

б) Ребро (v_r, v_1) не существует, следовательно, гамильтонов цикл не может быть получен.

В случае 1 и 2б следует прибегнуть к возвращению. Если нужны все гамильтоновы циклы, то в случае 2а обработать гамильтонов цикл и сделать шаг возвращения.

Возвращение состоит в удалении последней включенной вершины из S после чего S примет вид:

$$S = \{v_1, a \dots v_{r-1}\}$$

И добавление к S первой возможной вершины, следующего за v_r в столбце v_{r-1} .

Если не существует ни какой возможной вершины, то делается следующий шаг возвращения.

Поиск заканчивается тогда и только тогда, когда S состоит из одной вершины v_1 и не существует ни какой возможной вершины, которую можно было добавить в S , шаг возвращения делает S пустым.

Это значит, что все гамильтоновы циклы найдены.

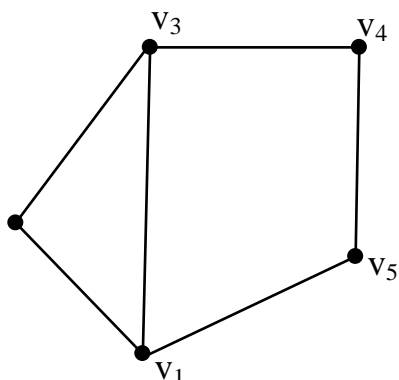
Алгоритм заканчивает работу.

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	0	1
v_2	1	0	1	0	0
v_3	1	1	0	1	0
v_4	0	0	1	0	1
v_5	1	0	0	1	0

Пример:

A_G – матрица смежности:

Граф G :



Множество S :

$$1) S = \{ v_1 \}$$

$$2) S = \{ v_1, v_2 \}$$

$$3) S = \{ v_1, v_2, v_3 \}$$

$$4) S = \{ v_1, v_2, v_3, v_4 \}$$

$$5) \underline{S = \{ v_1, v_2, v_3, v_4, v_5 \} - \Gamma}$$

$$6) S = \{ v_1, v_2, v_3, v_4 \}$$

$$7) S = \{ v_1, v_2, v_3 \}$$

$$8) S = \{ v_1, v_2 \}$$

$$9) S = \{ v_1 \}$$

$$10) S = \{ v_1, v_3 \}$$

$$11) S = \{ v_1, v_3, v_2 \}$$

Комментарии:

Выбираем начальную вершину графа v_1

Добавляем первую возможную вершину в столбце v_1 (т.е. вершину v_2)

Первая вершина (v_1) в столбце v_2 не является возможной, т.к. она уже принадлежит множеству S , поэтому добавляем следующую вершину в столбце (т.е. вершину v_3)

Добавляем вершину v_4

Добавляем вершину v_5 и видим, что это гамильтонова цепь. Дуга (v_5, v_1) дает гамильтонов цикл

Возвращение

Возвращение

Возвращение

Возвращение

Добавляем вершину v_3

Добавляем вершину v_2

- 12) $S = \{ v_1, v_3 \}$ В столбце v_2 нет возможной вершины.
Возвращение
- 13) $S = \{ v_1, v_3, v_4 \}$ Добавляем вершину v_4
- 14) $S = \{ v_1, v_3, v_4, v_5 \}$ Добавляем вершину v_5 . Дуга (v_5, v_1) дает цикл, но он не является гамильтоновым, т.к. во множестве S отсутствует вершина v_2
- 15) $S = \{ v_1, v_3, v_4 \}$ Возвращение
- 16) $S = \{ v_1, v_3 \}$ Возвращение
- 17) $S = \{ v_1 \}$ Возвращение
- 18) $S = \{ v_1, v_5 \}$ Добавляем вершину v_5
- 19) $S = \{ v_1, v_5, v_4 \}$ Добавляем вершину v_4
- 20) $S = \{ v_1, v_5, v_4, v_3 \}$ Добавляем вершину v_3
- 21) $S = \{ v_1, v_5, v_4, v_3, v_2 \} - \Gamma$ Добавляем вершину v_2 и видим, что это гамильтонова цепь. Дуга (v_2, v_1) дает гамильтонов цикл
- 22) $S = \{ v_1, v_5, v_4, v_3 \}$ Возвращение
- 23) $S = \{ v_1, v_5, v_4 \}$ Возвращение
- 24) $S = \{ v_1, v_5 \}$ Возвращение
- 25) $S = \{ v_1 \}$ Возвращение
- 26) $S = \emptyset$ Конец поиска

Задача коммивояжера и задача китайского почтальона

Задача коммивояжера – это обобщенная задача о поиске гамильтонова цикла в графе.

Задача: построить маршрут (цикл, цепь) во взвешенном графе G минимального веса, проходящий, по крайней мере, 1 раз по каждой вершине исходного графа.

Задача китайского почтальона – это обобщение задачи о поиске эйлерового цикла в графе.

Задача: построить маршрут (цикл, цепь), проходящий по каждому ребру графа, по крайней мере, 1 раз с минимальным суммарным весом.

Задание к лабораторной работе

Исходные графы G_1 : $(13, \{5, 6\})$

G_2 : $(7, \{3, 4\})$

1. Определить, является ли граф G_1 эйлеровым.

Если граф G_1 – эйлеров, то:

- построить эйлеров цикл по алгоритму Флёри;
- решить задачу «китайского почтальона», удалив минимальное число ребер, делающих его не эйлеровым (в качестве весов ребер взять 1).

Если G_1 не является эйлеровым, то:

- построить эйлеровы цепи в графе G_1 ;
- решить задачу «китайского почтальона» (в качестве весов ребер взять 1).
- добавить минимальное число ребер, делающих его эйлеровым и найти эйлеров цикл по алгоритму Флёри;

2. Определить, является ли граф G_2 гамильтоновым.

Если граф – гамильтонов, то:

- построить гамильтонов цикл, используя дерево полного перебора;
- построить гамильтоновы циклы, используя алгоритм Робертса-Флореса;
- решить для него задачу коммивояжера, удалив минимальное число ребер, нарушающих свойство гамильтоновости (в качестве весов ребер взять 1).

Если граф не является гамильтоновым, то:

- решить задачу коммивояжера (в качестве весов ребер взять 1);
 - добавить минимальное число ребер, делающих его гамильтоновым; построить гамильтонов цикл, используя дерево полного перебора и алгоритм Робертса-Флореса.
3. Привести примеры гамильтоновых графов, не удовлетворяющих теоремам Оре и Дирака.

Контрольные вопросы

1. Определение эйлера цикла, графа.
2. Сформулировать критерий существования в графе эйлера цикла.
3. Какой граф называется гамильтоновым? Дать определение гамильтонова цикла.
4. Сформулировать теоремы Оре, Дирака
5. Что находят задача китайского почтальона, задача коммивояжера?

Лабораторная работа № 6

Ориентированные графы или орграфы

Цель работы: изучение основных понятий для ориентированных графов, получение практических навыков нахождения сильных компонент, построения конденсации, базы и антибазы, нахождения ядра.

Теоретическая справка**Определение орграфа**

Ориентированный граф или орграф $G = (V, A)$ – пара множеств V и A таких, что V – конечное непустое множество, а A – некоторое подмножество декартового квадрата V :

$$A \subseteq V^2, V^2 = V \times V.$$

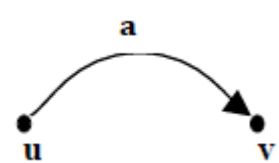
Вершины графа G – элементы множества V .

Дуги графа G – элементы множества A .

Дуга – упорядоченная пара вершин $a = (u, v)$.

Начало дуги – вершина u , **конец дуги** – вершина v .

Дуга **исходит** из своего начала и **заходит** в свой конец.



Орграф G – **орграф p -го порядка**, если мощность множества $|V| = p$.

Способы описания орграфов**1. Матрица смежности**

$$A = \| a_{ij} \|, i, j = \overline{1, p}, |V| = p, |A| = q,$$

$$a_{ij} = \begin{cases} 1, & (i, j) \in A; \\ 0, & (i, j) \notin A. \end{cases}$$

2. Матрица инцидентности

$$B = \| b_{ij} \|, i = \overline{1, p}, j = \overline{1, q}, |A| = q, |V| = p.$$

$$b_{ij} = \begin{cases} 1, & i, v \in V, j = (i, v) \in A; \\ -1, & i, v \in V, j = (v, i) \in A; \\ 0, & \text{else.} \end{cases}$$

Степени вершин орграфа

Полустепень захода вершины v графа G – число дуг, заходящих в вершину v

$$\deg^-(v) = |X|; \quad X = \{x \mid x = (u, v) \in A\}.$$

Полустепень исхода вершины v графа G – число дуг, исходящих из вершины v

$$\deg^+(v) = |Y|; \quad Y = \{y \mid y = (v, u) \in A\}.$$

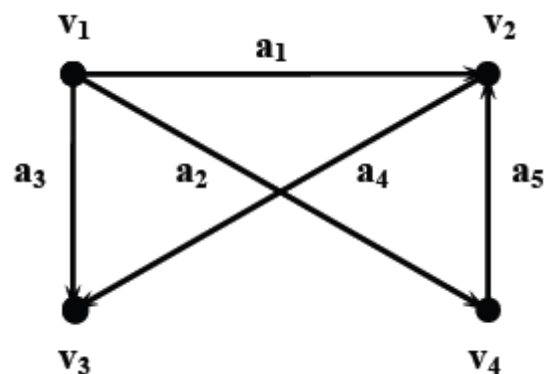
Степень вершины v орграфа G – сумма полустепеней захода и исхода вершины v : $\deg(v) = \deg^-(v) + \deg^+(v)$.

Лемма о рукопожатиях для орграфа

Сумма полустепеней захода всех вершин орграфа G равна сумме полустепеней исхода, и равна количеству дуг.

$$\sum_{i=1}^p \deg^+(v_i) = \sum_{i=1}^p \deg^-(v_i) = q, \quad q = |A|, \quad p = |V|.$$

Например: орграф $G = (V, A)$.



Матрица смежности A_G

	v_1	v_2	v_3	v_4	$\text{deg}^+(v_i)$
v_1	0	1	1	1	3
v_2	0	0	1	0	1
v_3	0	0	0	0	0
v_4	0	1	0	0	1
$\text{deg}^-(v_i)$	0	2	2	1	

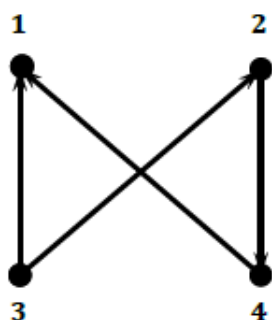
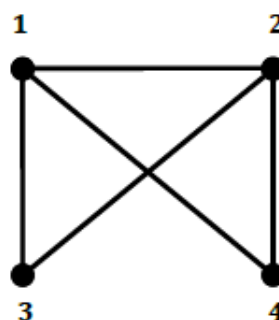
Матрица инцидентности B_G

	a_1	a_2	a_3	a_4	a_5
v_1	1	1	1	0	0
v_2	-1	0	0	1	-1
v_3	0	0	-1	-1	0
v_4	0	-1	0	0	1

Основание – неориентированный граф, получившийся в результате снятия ориентации дуг орграфа.

Обратный граф $G^{-1} = (V, A^{-1})$ – орграф, у которого множество вершин совпадает с исходным графом, и дуга

$$(u, v) \in A^{-1} \Leftrightarrow (v, u) \in A.$$

Обратный орграф G^{-1} Основа G

Маршруты в орграфах

Ориентированный маршрут (ормаршрут) – конечная чередующаяся последовательность вершин и дуг орграфа таких, что каждая дуга исходит из предыдущей вершины ормаршрута и заходит в последующую вершину $a_i = (v_i, v_{i+1})$.

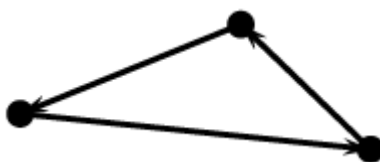
Ориентированная цепь (орцепь) – ориентированный маршрут без повторяющихся дуг.

Путь – цепь без повторяющихся вершин.

Ориентированный цикл – замкнутая ориентированная цепь.

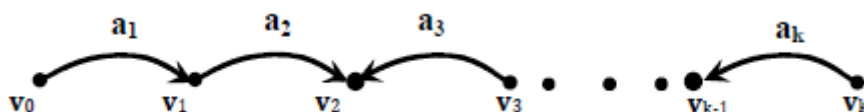
Контур – замкнутый путь или замкнутый маршрут без повторения дуг и вершин (кроме концевых).

Пример контура:



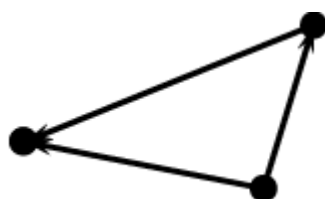
Длина ориентированного маршрута – число дуг, составляющих маршрут, с учетом повторения.

Полумаршрут (маршрут основания) – последовательность вершин и дуг орграфа, что: $a_i = (v_{i+1}, v_i)$ или $a_i = (v_i, v_{i+1})$.



Аналогично вводятся понятия **полуцепь, полупуть, полуконтур**.

Пример полуконтура:



Типы связности орграфа

Вершина v графа G **достижима** из вершины u , если существует (u, v) -маршрут (путь) в G : $u \rightarrow \dots \rightarrow v$, соответственно вершина u – **контрдостижима** для вершины v .

Любая вершина считается **достижима для самой себя**.

Вершины v и u графа G – **взаимнодостижимы**, если вершина v достижима для вершины u , и вершина u достижима для вершины v : $u \rightarrow \dots \rightarrow v$ и $v \rightarrow \dots \rightarrow u$.

Орграф G – сильносвязный (сильный), если любые две вершины в нём взаимнодостижимы.

Орграф G – одностороннесвязный (односторонний), если для каждой пары его вершин, по крайней мере, одна достижима из другой.

Орграф G – слабосвязный (слабый), если любые две его вершины соединены полумаршрутом (полупутем).

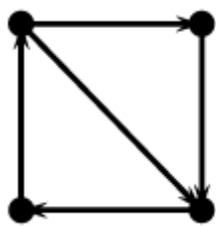
Орграф G – несвязен, если не связно его основание.

Сильная компонента – максимальный относительно включения вершин сильный подграф исходного орграфа.

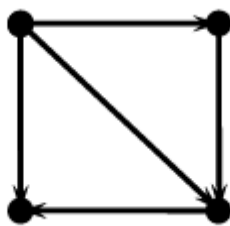
Односторонняя компонента – максимальный относительно включения односторонний подграф исходного орграфа.

Слабая компонента – максимальный относительно включения слабый подграф исходного орграфа.

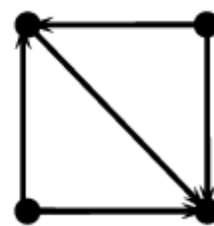
Например:



Сильный орграф



Односторонний орграф



Слабый орграф

Критерии сильной, односторонней и слабой связности

Орграф является **сильным** тогда и только тогда, когда в нём есть остовный циклический маршрут

Орграф является **односторонним** тогда и только тогда, когда в нём есть остовный маршрут

Орграф является **слабым** тогда и только тогда, когда в нём есть остовный полумаршрут

Остовный маршрут – маршрут, содержащий все вершины исходного графа.

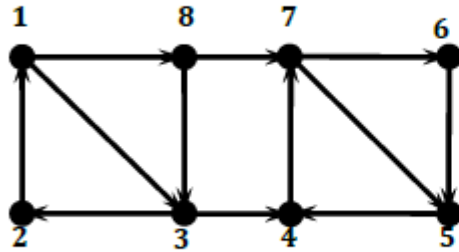
Конденсация орграфа

Конденсация орграфа G – орграф G^* , вершины S_1, S_2, \dots, S_m которого соответствуют сильным компонентам орграфа G , и дуга (S_i, S_j) принадлежит орграфу G^* тогда и только тогда, когда в G

существует дуга, начало которой находится в сильной компоненте S_i , конец – в S_j .

Конденсация G^* любого орграфа не имеет контуров

Например:

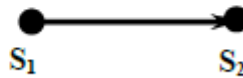


Орграф G

Сильные компоненты G

$$S_1 = \{1, 2, 3, 8\}$$

$$S_2 = \{4, 5, 6, 7\}$$



Орграф G^* – конденсация G

Алгоритм построения конденсации

1. Построим матрицу достижимости орграфа $G = (V, A)$:

$$R = \|r_{ij}\|, i, j = \overline{1, p},$$

$$r_{ij} = \begin{cases} 1, & j \rightarrow \dots \rightarrow i; \\ 0, & \text{else.} \end{cases}$$

2. Построить матрицу контрдостижимости орграфа G :

$$Q = \|q_{ij}\|, i, j = \overline{1, p}, Q = R^T,$$

$$q_{ij} = \begin{cases} 1, & i \rightarrow \dots \rightarrow j; \\ 0, & \text{else.} \end{cases}$$

4. Найдем матрицу взаимной достижимости, где “ \otimes ” – оператор поэлементного умножения матриц:

$$S = R \otimes Q = R \otimes R^T, s_{ij} = r_{ij} \times q_{ij}, i, j = \overline{1, p}.$$

4. Выберем некоторую вершину $v_i \in V$, тогда сильная компонента орграфа, содержащая вершину v_i , определяется единичными элементами i -той строки матрицы S . Иначе: перестановкой строк и столбцов можно привести матрицу S

Матрица взаимной достижимости S_G

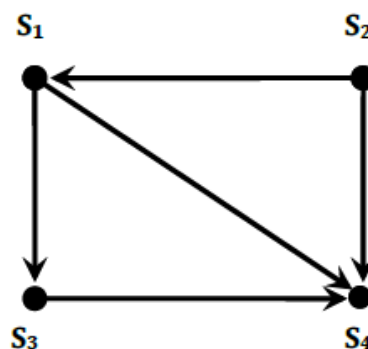
	1	2	3	4	5	6	7	8
1	1	1	0	1	1	0	0	0
2	1	1	0	1	1	0	0	0
3	0	0	1	0	0	0	0	0
4	1	1	0	1	1	0	0	0
5	1	1	0	1	1	0	0	0
6	0	0	0	0	0	1	1	0
7	0	0	0	0	0	1	1	0
8	0	0	0	0	0	0	0	1

Блочно-диагональный вид
матрицы взаимной достижимости S_G

	1	2	4	5	3	6	7	8
1	1	1	1	1	0	0	0	0
2	1	1	1	1	0	0	0	0
4	1	1	1	1	0	0	0	0
5	1	1	1	1	0	0	0	0
3	0	0	0	0	1	0	0	0
6	0	0	0	0	0	1	1	0
7	0	0	0	0	0	1	1	0
8	0	0	0	0	0	0	0	1

Сильные компоненты G : $S_1 = \{1,2,4,5\}$, $S_2 = \{3\}$, $S_3 = \{6,7\}$, $S_4 = \{8\}$.

Орграф G^* - конденсация орграфа G :

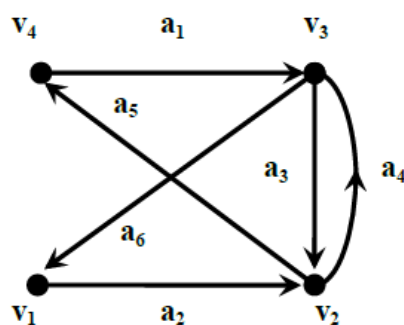


Обходы орграфа

Эйлеров цикл – цикл, содержащий каждую дугу орграфа.

Эйлеров орграф – связный орграф, содержащий эйлеров цикл.

Например:



Эйлеров цикл орграфа: $(a_1, a_3, a_4, a_6, a_2, a_5)$.

Критерий эйлеровости для орграфов

Для связного орграфа следующие условия эквивалентны:

- 1) орграф G – эйлеров;
- 2) для любой вершины справедливо: $\deg^+(v) = \deg^-(v)$.

Следствие из критерия эйлеровости для орграфов

Орграф G является объединением контуров, попарно не имеющих общих ребер.

Гамильтонов контур орграфа G – контур, содержащий все вершины данного орграфа.

Гамильтонов орграф G – орграф, содержащий гамильтонов контур.

База

База орграфа G – наименьшее (относительно включения) подмножество вершин V , удовлетворяющее условию: любая вершина $v \in V/V$ достижима из какой-либо вершины $u \in V$.

Базовая компонента – сильная компонента орграфа G , в которую не входит ни одна дуга из других сильных компонент.

В конденсации G^* таким компонентам соответствуют вершины с нулевыми полустепенями захода.

Подмножество вершин V орграфа G – база, если V состоит из вершин принадлежащих базовым компонентам, причем в каждую базовую компоненту входит ровно одна вершина из множества V .

База определяется не единственным образом, исключая ациклический или бесконтурный граф.

В любом орграфе существует база, никакие две вершины базы не соединены ормаршрутом.

Вершины – полустепени захода, которые равны 0, принадлежат базе.

База бесконтурного орграфа состоит только из вершин, полустепени захода которых равны нулю.

Алгоритм нахождения базы

1. Построить конденсацию G^* .
2. Выделить в конденсации вершины с нулевыми полустепенями захода. Такие вершины будут определять базовые компоненты.
3. Из каждой базовой компоненты выбирается по одной вершине, таким образом, база орграфа может быть определена не единственным образом.

Антибаза

Антибаза орграфа G – наименьшее (относительно включения) подмножество вершин V' , удовлетворяющее условию: любая вершина $v \in V$, достижима из какой-либо вершины $u \in V/B'$.

Алгоритм построения антибазы

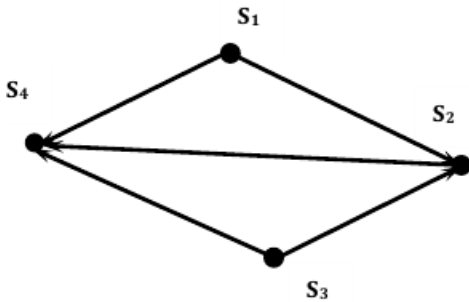
1. Построить конденсацию G^* .
2. Выделить в конденсации вершины с нулевыми полустепенями исхода.
3. Из каждой компоненты, соответствующей такой вершине, выбирается по одной вершине.

Например.

Сильные компоненты орграфа G :

$$S_1 = \{1, 2, 3\}, S_2 = \{6\}, S_3 = \{4, 5\}, S_4 = \{7, 8\}$$

Конденсация G^* орграфа G :



Для определения базы: выделим в конденсации вершины с нулевыми полустепенями захода. Базовые компоненты: S_1 и S_3 . Базы орграфа G : $\{1, 4\}$; $\{1, 5\}$; $\{2, 4\}$; $\{2, 5\}$; $\{3, 4\}$; $\{3, 5\}$.

Для антибазы: выделить в конденсации вершины с нулевыми полустепенями исхода. Антибазовая компонента: S_4 . Антибаза орграфа G : $\{7\}$; $\{8\}$.

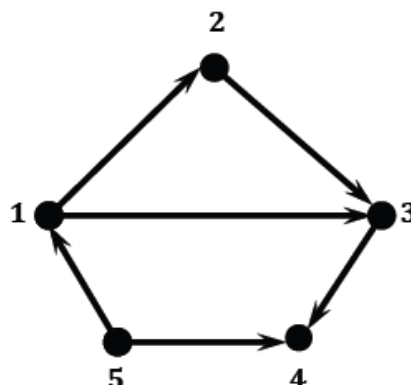
Ядро графа

Доминирующее множество вершин S графа $G = (V, A)$ – подмножество вершин такое, что для любой вершины $w \in V - S$ существует такая вершина $v \in S$, что $(v, w) \in A$.

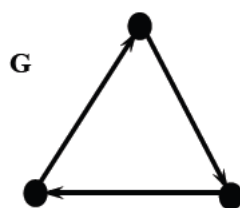
Независимое множество вершин графа G – подмножество вершин графа G , в котором никакие две вершины не смежны между собой.

Ядро графа – множество вершин, которое одновременно является доминирующим и независимым множеством.

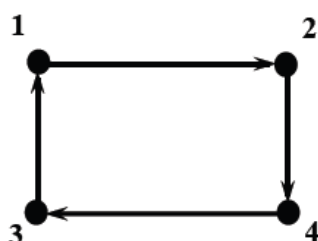
Например: Множество вершин $\{2, 5\}$ - ядро орграфа.



Пример орграфа, в котором ядра не существует.



Пример орграфа, в котором два ядра: $\{1, 4\}$, $\{2, 3\}$.



Задание к лабораторной работе

1. Используя алгоритм генерации варианта GV , построить ориентированный граф G : $GV(9, \{3, 4\})$. Ориентацию дуг установить произвольно.
2. Построить матрицу смежности и матрицу инцидентности заданного орграфа. Проверить выполнение леммы о рукопожатиях для орграфа.
3. Построить основание и обратный граф. Определить является ли граф симметричным.
4. Построить, если это возможно, ормаршрут (не орцепь), выделить в нем цепь (не путь), путь, замкнутый маршрут (не цикл), цикл (не контур) и контур, полумаршрут (не полуцепь), полуцепь (не полупуть), полупуть. Обосновать отсутствие какого-либо из маршрутов.
5. Построить матрицу достижимости, контрдостижимости, взаимной достижимости.
6. Определить тип связности орграфа, выделить сильные компоненты.
7. Построить конденсацию. Определить базы и антибазы.
8. Выделить ядро орграфа.
9. Найти в графе контур Эйлера и Гамильтона.

10. Привести пример орграфа, в котором $m+1$ баз мощностью $n+1$, где n – первая цифра порядкового номера студента в списке группы, m – соответственно, вторая цифра в номере.

Контрольные вопросы

1. Определение орграфа.
2. Дуги орграфа. Начало и конец дуги.
3. Матричные способы описания орграфов и их особенности. Матрица смежности и матрица инцидентности.
4. Основание и обратный орграф. Симметричный орграф.
5. Полуостепенность исхода и полуостепенность захода вершин орграфа. Степень вершины орграфа.
6. Лемма о рукопожатиях для орграфа.
7. Ориентированный маршрут, ориентированная цепь, путь.
8. Ориентированный замкнутый маршрут, ориентированный цикл и контур.
9. Полумаршрут, полуцепь, полупуть, полувцикл, полуконтур. Длина ормаршрута.
10. Достижимость, контрдостижимость и взаимная достижимость.
11. Сильная, односторонняя и слабая связность. Компоненты связности.
12. Критерии сильной, односторонней и слабой связности орграфа. Остовный маршрут.
13. Конденсация орграфа и алгоритм ее построения. Свойства конденсации.
14. База, антибаза и алгоритмы их построения. Базовая компонента.
15. Дать определение ядра графа.
16. Гамильтонов контур и орграф.
17. Эйлеров цикл и орграф.

Лабораторная работа № 7

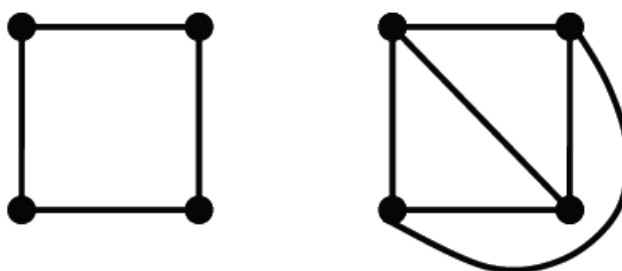
Плоские и планарные графы

Цель работы: приобретение практических навыков в определении планарности графов на основе критериев Понтрягина-Куратовского и Вагнера, построении плоской укладки и определении числовых характеристик непланарных графов.

Теоретическая справка**Плоские и планарные графы**

Плоским называется такой граф G , у которого вершины – точки плоскости, а ребра – непрерывные плоские линии без пересечений и самопересечений, причем соединяющие вершины так, что никакие два ребра не имеют общих точек, кроме инцидентных им обоим вершин.

Пример:



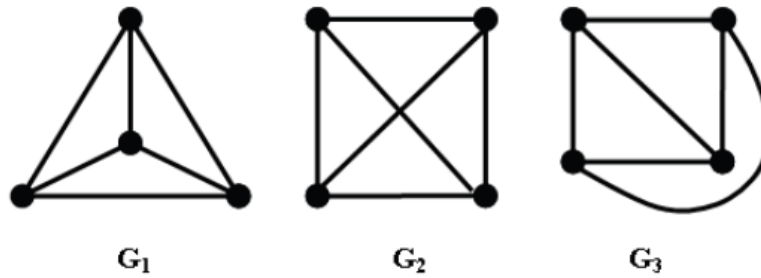
Планарный граф – это граф, который изоморфен плоскому.

О планарных графах говорят, что они имеют **плоскую укладку** или **укладываются на плоскости**.

Утверждение 1. Всякий подграф планарного графа – планарен.

Утверждение 2. Если некоторый граф содержит непланарный подграф, то и сам граф не планарен.

На рисунке приведено три изображения графа K_4 .



Графы G_1 , G_3 являются плоскими по определению, а граф G_2 – планарен, так как изоморфен плоскому графу.

Теорема Жордано

Жорданова кривая – это непрерывная спрямляемая линия, не имеющая самопересечений.

Теорема Жордано

Замкнутая Жорданова кривая L на плоскости делит область на две области, так, что любая линия, соединяющая точки в различных подобластях пересекает Жорданову кривую



Гранью плоского графа называется максимальное по включению количество точек плоскости, каждая пара которых может быть соединена Жордановой кривой, не пересекающей ребра графа.

Граница грани – это множество вершин и ребер, принадлежащих грани.

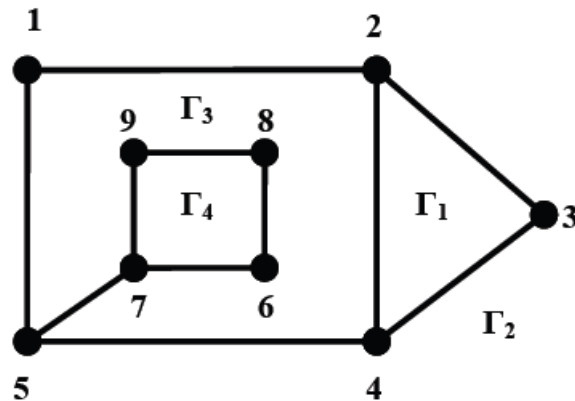
Всякий плоский граф имеет одну единственную неограниченную **внешнюю** грань, остальные – **внутренние**.

Следствие из теоремы Жордано

Любые две вершины, принадлежащие одной грани, могут быть соединены цепью произвольной длины таким образом, что выбранная грань разбивается на две грани

Например

Грани $\Gamma_1, \Gamma_3, \Gamma_4$ – внутренние, грань Γ_2 – внешняя.



Границы граней:

$\Gamma_1 = \{2,3,4\}$ или $\{23, 24, 34\}$;

$\Gamma_2 = \{1, 2, 3, 4, 5\} \rightarrow \{12, 23, 34, 45, 15\}$;

$\Gamma_3 = 1)\{1, 2, 4, 5\} \rightarrow \{12,15, 24, 45\}$;

2) $\{5, 6, 7, 8, 9\} \rightarrow \{57, 67, 68, 79, 89\}$;

$\Gamma_4 = \{6,7,8,9\} \rightarrow \{67,68,79,89\}$.

Теорема Эйлера для плоского графа

Для любого связного графа $G = (V, E)$, $|V| = p$, $|E| = q$,
являющегося плоским, справедливо соотношение:

$$p - q + f = 2,$$

где p – количество вершин, q – количество ребер,

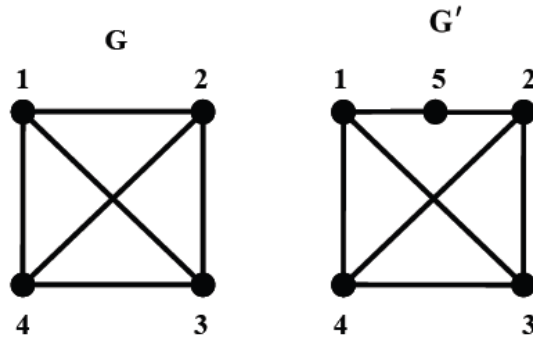
f – количество граней плоского графа

Графы K_5 и $K_{3,3}$ – непланарны.

Критерии планарности графов

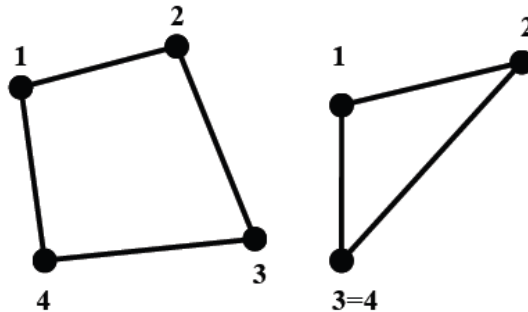
Операция подразбиения ребер

В произвольном графе G удаляется ребро $e = \{u, v\}$ и добавляется два новых ребра: $e_1 = \{u, a\}$, $e_2 = \{a, v\}$, где a – новая вершина, не принадлежащая графу. Таким образом, ребро $\{u, v\}$ графа G подразбивается вершиной a на два ребра и получается новый граф : $G' = G - \{u, v\} + \{u, a\} + \{a, v\}$.

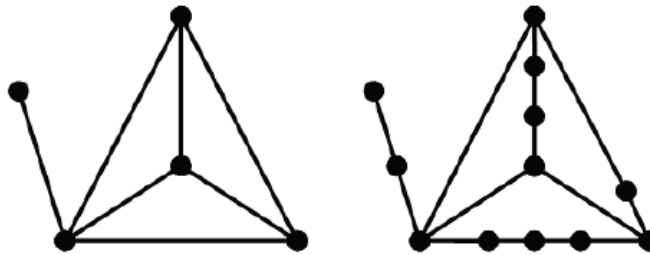


Операция стягивания ребер

Операция стягивания ребер – отождествление смежных вершин ребра (операция противоположная операции подразделения ребер).



Два графа называются **гомеоморфными**, если они могут быть получены из одного и того же графа путем подразделения его ребер.



Критерий планарности Понтрягина-Куратовского

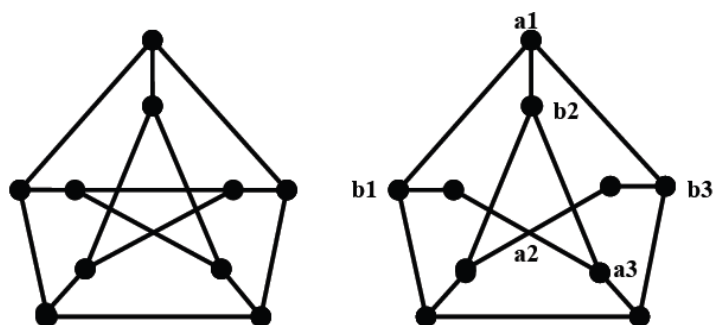
Граф планарен тогда и только тогда, когда он не содержит подграфов гомеоморфных K_5 или $K_{3,3}$

Критерий планарности Вагнера

Граф планарен тогда и только тогда, когда он не содержит подграфов, стягиваемых к графам K_5 или $K_{3,3}$

Например.

Граф Петерсона стягивается к K_5 , поэтому – непланарен. Граф справа от графа Петерсона стягивается к $K_{3,3}$, поэтому также непланарен.



Алгоритм плоской укладки графа

Для плоской укладки графа и проверки является ли он планарным, используется алгоритм γ .

Для правильной работы алгоритма (без ограничения области применения) определим свойства графов, которые подаются на вход алгоритма:

- граф должен быть связным;
- граф должен иметь хотя бы один цикл;
- граф не должен содержать мостов, т.е. ребер после удаления которых граф распадается на несколько компонент связности.

Алгоритм γ плоской укладки графа G представляет собой процесс последовательного присоединения к некоторому уже уложенному на плоскости графу $!G$ (подграф графа G) некоторой новой цепи также принадлежащей G , оба конца которой принадлежат $!G$. Эта цепь разбивает одну из граней графа $!G$ на две.

При этом в качестве начального плоского графа $!G$ выбирается любой простой цикл исходного графа. Процесс продолжается до тех пор, пока не будет получена плоская укладка графа G или присоединение некоторой цепи оказывается невозможным, в том случае граф является не планарным.

Пусть есть граф G и построена некоторая плоская укладка $!G$ подграфа графа G .

Сегментом S относительно текущей плоской укладки $!G$ или просто **сегментом** будем называть подграф исходного графа G одного из следующих двух видов:

- 1) ребро $e = \{u, v\}$ исходного графа G такое, что не принадлежит текущей плоской укладке графа, $e \notin \tilde{G}$, но концевые вершины этого ребра u, v принадлежат этой плоской укладке;
- 2) связная компонента графа $G - \tilde{G}$ дополненная всеми ребрами графа G , инцидентными вершинам взятой компоненты и концевыми вершинами этих ребер.

Граф $G - \tilde{G}$ получается вычитанием графа $!G$ из исходного графа G .

Контактная вершина – это вершина v сегмента S относительно $!G$, которая принадлежит множеству вершин текущей плоской укладки.

Допустимой гранью для сегмента S называется такая грань Γ графа $!G$, которая содержит все контактные вершины сегмента S .

Обозначим $\Gamma(S_i)$ – множество всех допустимых граней для сегмента S_i .

Простая цепь α сегмента S , содержащая 2 различные контактные вершины и не содержащая других контактных вершин, называется α -цепью.

Простая α -цепь проходит из контактной вершины через неконтактные и возвращается в контактную вершину.

Алгоритм γ .

0. Выберем некоторый простой цикл C графа G и уложим его на плоскости (лучше выбирать простой цикл графа G , доставляющий окружение графа): $\tilde{G} := C$.

1. Найдем грани \tilde{G} графа и множество сегментов S относительно текущей укладки \tilde{G} . Если множество сегментов пусто перейти к п. 7.

2. Для каждого сегмента S найдем множество допустимых граней $\Gamma(S)$.

3. Если существует сегмент S , для которого $\Gamma(S) = \emptyset$ (множество граней пусто), то граф G не планарен. Выход из алгоритма, иначе переход к п. 4.

4. Если существует сегмент S для которого ровно одна допустимая грань ($\Gamma(S) = 1$), то перейдем к п. 6, иначе п. 5.

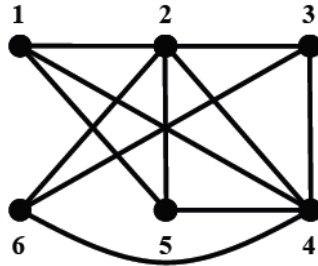
5. Для некоторого сегмента S выбираем произвольную допустимую грань Γ

6. Поместим произвольную α -цепь L , принадлежащую S , в грань Γ , заменим \tilde{G} на $\tilde{G} \cup L$ и перейдем к п. 1. Построена \tilde{G} частичная, текущая плоская укладка графа G .

7. Построена \tilde{G} плоская укладка графа G .

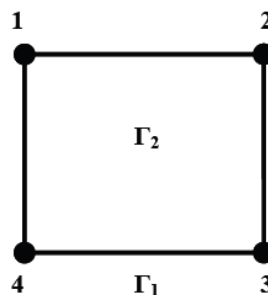
Пример.

Определить планарность и построить плоскую укладку графа G .



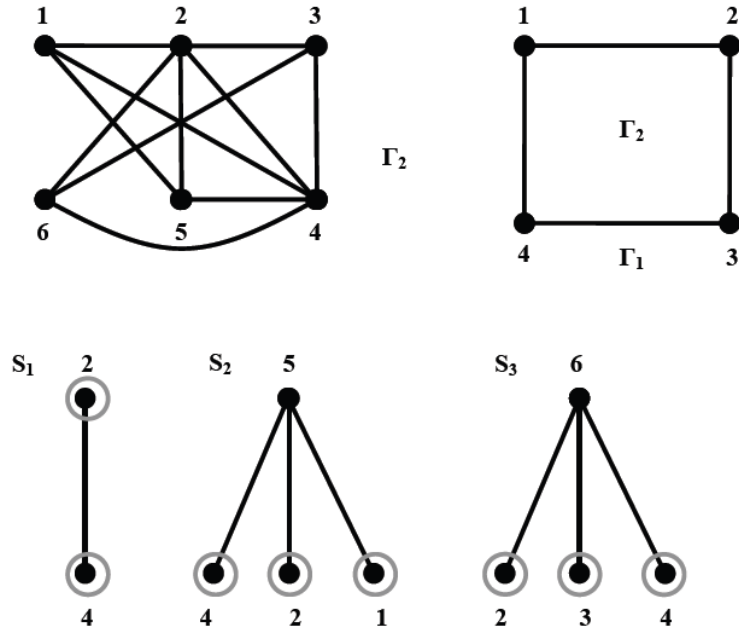
Инициализация алгоритма:

- выберем в графе произвольный простой цикл и уложим его на плоскости $\tilde{G} := C$;
- определим для него множество граней.

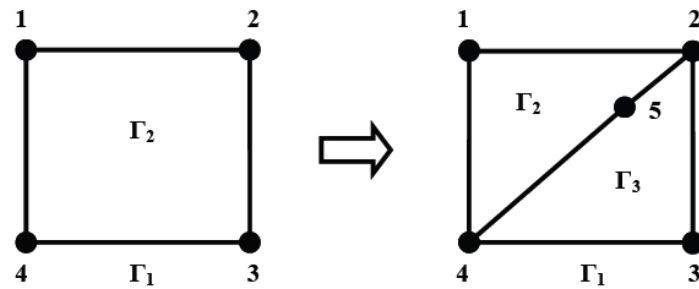


1) Определим множество сегментов относительно текущей плоской укладки. Множество сегментов не пусто.

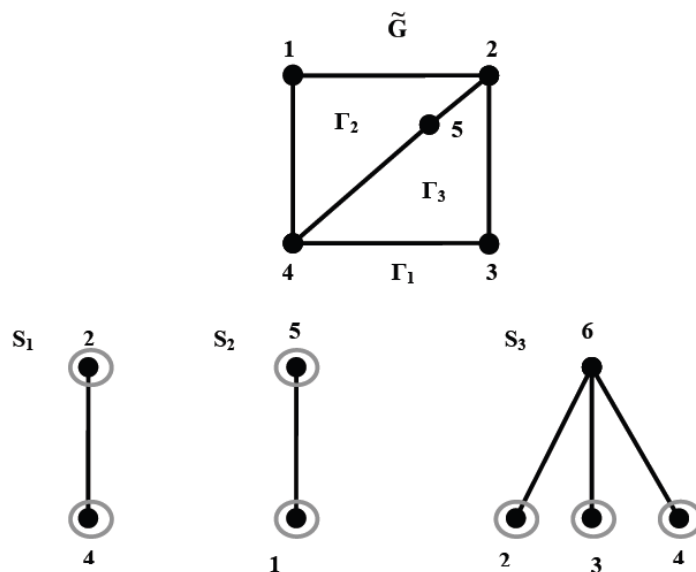
2) Определим для каждого сегмента множество допустимых граней:
 $\Gamma(S1) = \Gamma(S2) = \Gamma(S3) = \{\Gamma1, \Gamma2\}$.



3) Выберем сегмент S_2 и α -цепь = $\{4, 5, 2\}$, уложим ее в одной из допустимых граней, пусть это будет Γ_2 .



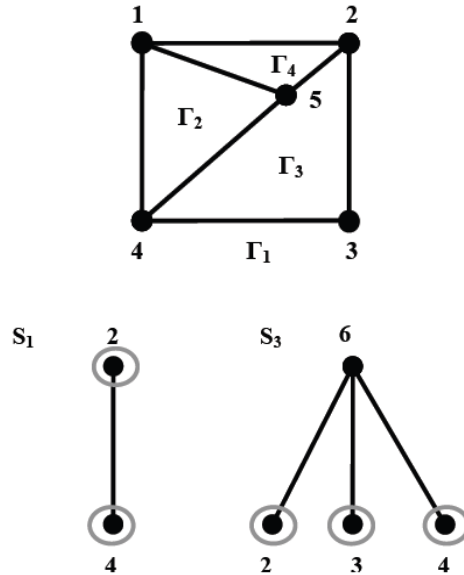
4) Для новой текущей плоской укладки определяем новые множества граней и сегментов.



5) Для каждого сегмента определим множества допустимых граней: $\Gamma(S_1) = \{\Gamma_1, \Gamma_2, \Gamma_3\}$, $\Gamma(S_2) = \{\Gamma_2\}$, $\Gamma(S_3) = \{\Gamma_1, \Gamma_3\}$. Выбираем сегмент S_2 и

укладываем его в единственную для него допустимую грань Γ_2 . Получаем текущую частичную плоскую укладку графа G .

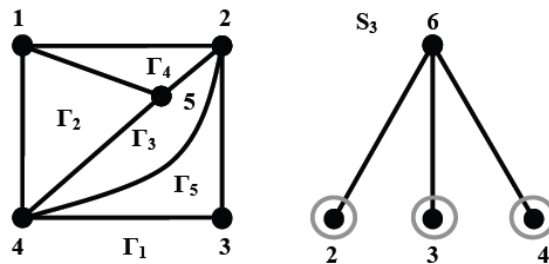
б) Для новой текущей частичной укладки $G \sim$ определим множество граней и сегментов.



Множества допустимых граней для сегментов: $\Gamma(S_1) = \Gamma(S_3) = \{\Gamma_1, \Gamma_3\}$.

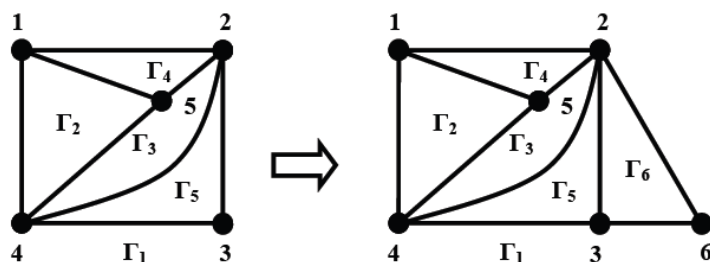
Выбираем сегмент S_1 и укладываем в допустимой для него грани Γ_3 .

7) Получаем новую частичную укладку. Определяем для нее множество граней и сегментов.

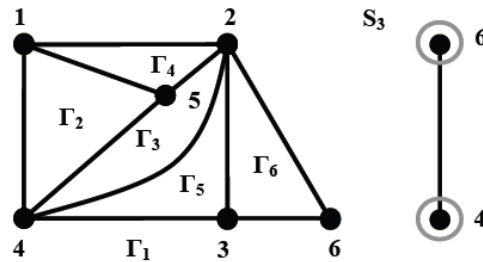


Множество допустимых граней для сегмента $\Gamma(S_3) = \{\Gamma_1, \Gamma_5\}$.

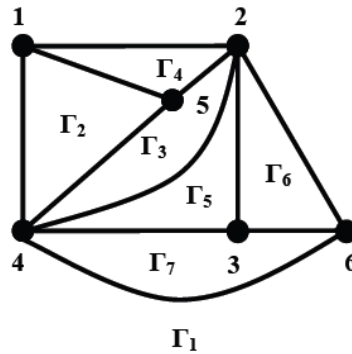
8) Выбираем грань Γ_1 и α -цепь = $\{3, 6, 2\}$, укладываем его в допустимой грани и получаем новую текущую частичную укладку.



9) Для новой текущей частичной укладки находим множество граней и сегментов. Для сегмента S_3 множество допустимых граней составляет: $\Gamma(S_3) = \{\Gamma_1\}$. Закljučаем сегмент S_3 до допустимой грани Γ_1 .



10) Получаем новую текущую плоскую укладку. Множество сегментов пустая, следовательно, алгоритм закончил работу. Входной граф -планарный и построена его плоская укладка.



Характеристики не планарных графов

Число скрещиваний графа G – это \min число пересечений двух ребер при изображении графа G на плоскости (обозначают $Cr(G)$).

Число скрещиваний равно 0, если граф планарен.

Искаженность G – это минимальное число ребер, удаление которых приводит к планарному графу (обозначают $Sk(G)$).

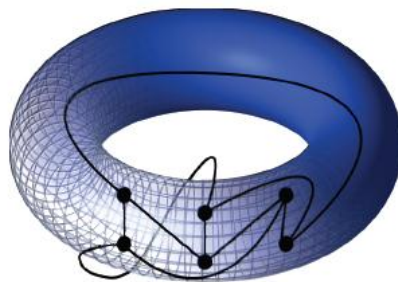
Толщина G – это минимальное число его планарных подграфов, объединение которых дает исходный граф G (обозначают $t(G)$).

Род графа G – это минимальное число ручек, которые необходимо добавить к сфере, чтобы можно было уложить граф G без пересечений, самопересечений ребер.

Непланарный граф, укладываемый на торе без пересечений и самопересечений ребер называются **торидальными**, род такого графа равен 1.

К тороидальным графам относят графы K_5 , K_7 , $K_{3,3}$, $K_{4,4}$.

Пример укладки графа $K_{3,3}$ на торе:



Задание к лабораторной работе

Исходные данные граф $G:GV(13, \{5, 6\})$.

1. Определить, является ли исходный граф G планарным или непланарным, используя критерий Понтрягина-Куратовского или Вагнера.

Найти подграф G , гомеоморфный K_5 или $K_{3,3}$ по критерию Понтрягина-Куратовского или подграф, стягиваемый к K_5 или к $K_{3,3}$ по критерию Вагнера.

2. Если исходный граф планарен, обозначить его $G1$.

3. Если исходный граф непланарен, обозначить его $G2$.

4. Если исходный граф был планарен, добавить минимальное число ребер до непланарности и обозначить полученный непланарный граф $G2$.

5. Если исходный граф был непланарен, удалить минимальное число ребер и обозначить полученный планарный граф $G1$.

6. Количество добавляемых (удаляемых) при преобразованиях графа ребер должно быть обосновано.

7. Построить плоскую укладку графа $G1$, используя алгоритм γ . Продемонстрировать пошаговое выполнение алгоритма γ .

8. Для непланарного графа $G2$ найти род, толщину, искаженность и число скрещиваний.

Контрольные вопросы

1. Какой граф называется плоским, планарным?

2. Что такое жорданова кривая? Сформулировать теорему Жордано и следствие из нее.

3. Дать определение грани и границы грани.
4. Какие грани называют внутренними? внешними?
5. Сформулировать теорему Эйлера для плоского графа.
6. Операции подразбиения ребер и стягивания вершин.

Гомеоморфные графы.

7. Сформулировать критерии планарности Понтрягина - Куратовского и Вагнера.
8. Алгоритм плоской укладки графа. Определение сегмента, контактной вершины, α – цепи, допустимой грани.
9. Характеристики непланарных графов, род, толщина, число скрещиваний и искаженность.

Лабораторная работа № 8

Раскраска графов

Цель работы: приобретение практических навыков определения хроматического числа и индекса для неорграфов, построении оптимальной и субоптимальной правильной вершинной и реберной раскраски графов.

Теоретическая справка**Вершинная раскраска графов**

$G = (V, E)$ – простой неориентированный граф, k – натуральное число.

Вершинной k -раскраской или просто **k -раскраской** графа G называется произвольная функция f , отображающая множество вершин графа G в некоторое k -элементное множество:

$$f: VG \rightarrow \{a_1, a_2, \dots, a_k\} = A.$$

Если для некоторой вершины v графа G : $f(v)=i$, то говорят что вершина v **раскрашена в i -тый цвет**.

Раскраска называется **правильной**, если $f(u) \neq f(v)$ для любых смежных вершин u и v графа G (или концевые вершины любого ребра окрашены в разные цвета).

Граф, для которого существует правильная k -раскраска, называется **k -раскрашиваемым**.

Хроматическое число графа G – это минимальное число красок, при котором граф имеет правильную раскраску.

Если хроматическое число равно k , то граф называется **k -хроматическим** (обозначают $\chi(G) = k$).

Правильную k -раскраску графа G можно рассматривать как разбиение множества вершин графа G на не более чем k непустых множеств, которые называются **цветными классами**.

Графы с малым хроматическим числом

Лемма о 2-х раскрашиваемых графах

Пусть G – простой неориентированный граф:

1) $\chi(G) = 1$ тогда и только тогда, когда G – пустой граф, $\chi(O_p) = 1$.

2) $\chi(G) = 2$ тогда и только тогда, когда G – непустой двудольный граф.
Если непустой граф является деревом, то $\chi(G) = 2$.

Лемма о раскраске циклов

Хроматическое число всякого цикла, содержащего p вершин, равно 2, если p – четно, и 3, если p – нечетно.

Если граф G содержит цикл нечетной длины, то $\chi(G) > 2$.

Лемма о раскраске полного графа

Хроматическое число полного графа K_p равно p .

Если граф G содержит подграф изоморфный графу K_p , то $\chi(G) \geq p$.

Граф, у которого $\chi = 2$, называются **бихроматическим**.

Теорема Кёнига

Непустой граф является **бихроматическим** тогда и только тогда, когда он не содержит циклов нечетной длины.

Следствие 1. Любое дерево бихроматично.

Следствие 2. Любой двудольный граф бихроматичен.

Оценки хроматического числа графа

Под **нижними оценками хроматического числа** понимают неравенства вида: $\chi(G) \geq c$, где c – некоторая константа, вычисляемая по графу G , а под **верхними оценками** – неравенства вида $\chi(G) \leq c$, где c имеет тот же смысл.

Первая нижняя оценка

Для произвольного графа $G = (V, E)$, $|V| = p$, $|E| = q$
справедливо неравенство $\chi(G) \geq \frac{p^2}{p^2 - 2q}$

Хроматическое число и плотность графа или вторая нижняя оценка

Для произвольного графа G справедливо неравенство $\chi(G) \geq \varphi(G)$, где $\varphi(G)$ – плотность графа или кликовое число

Теорема о графах без треугольников

Для произвольного $k \geq 2$ существует простой связный граф G_k такой, что справедливо $\varphi(G_k) = 2$ и $\chi(G_k) = k$

Хроматическое число и число независимости графа или третья нижняя оценка

Для произвольного графа G справедливо неравенство: $\chi(G) \geq \frac{p}{\alpha(G)}$, где $\alpha(G)$ – число независимости графа

Верхние оценки хроматического числа

Для произвольного графа G справедливо неравенство: $\chi(G) \leq \Delta(G) + 1$, где $\Delta(G)$ – максимум из степеней вершин графа

Теорема Брукса

Для связного неполного графа G при условии, что $\Delta(G) \geq 3$, справедливо неравенство: $\chi(G) \leq \Delta(G)$.

Замечание о компонентах связности

Хроматическое число графа равно максимуму из хроматических чисел его компонент связности

Гипотеза о четырех красках

Хроматическое число любого планарного графа не превосходит 4

Теорема Хивуда

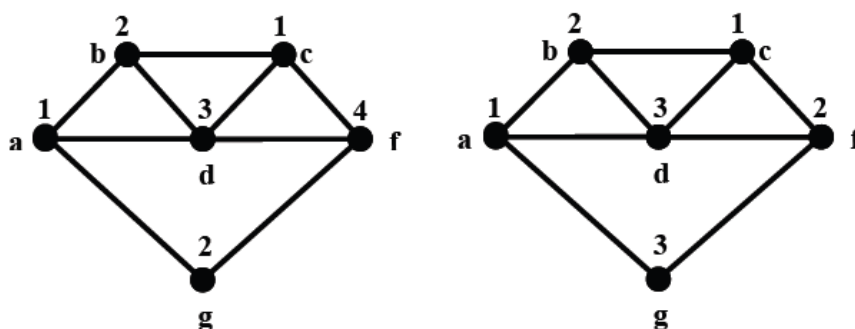
Хроматическое число любого планарного графа не превосходит 5

Алгоритм последовательной раскраски (субоптимальный)

1. Произвольной вершине графа G приписываем цвет 1.
2. Пусть раскрашены i вершин графа G в цвета от 1 до k , где $k \leq i$. Произвольной неокрашенной вершине v_{i+1} приписываем минимальный цвет, неиспользованный при раскраске смежных с ней вершин. Алгоритм последовательной раскраски зависит от способа выбора вершин на обслуживание.

Например.

В первом случае последовательность выбора вершин графа для раскраски такова: **(a, b, g, d, c, f)**. Число красок, использованных для правильной раскраски вершин графа, равно 4.



Во втором последовательность выбора вершин графа для раскраски: **(a, b, d, c, f, g)**. Число красок, использованных для правильной раскраски вершин графа, равно 3.

Последовательная раскраска вершин графа $G=(V,E)$, $|V|=p$, $|E|=q$, с матрицей $A_G = \|a_{ij}\|, i, j = \overline{1, p}$, смежности основанная на методах переупорядочения вершин.

1. «Наибольшие – первыми» или НП-упорядочение

Упорядочиваем вершины графа G в порядке невозрастания их степеней $\deg(v_i)$. Раскрашиваем вершины графа G по методу последовательной раскраски, выбирая вершины из этого списка. Если две вершины имеют одинаковые степени, то вычисляем двухшаговые степени вершины $\deg^2(v_i)$,

как число маршрутов длины 2, исходящих из вершины v_i и так далее. Рекуррентная формула для определения k – шаговой степени вершины графа по матрице смежности:

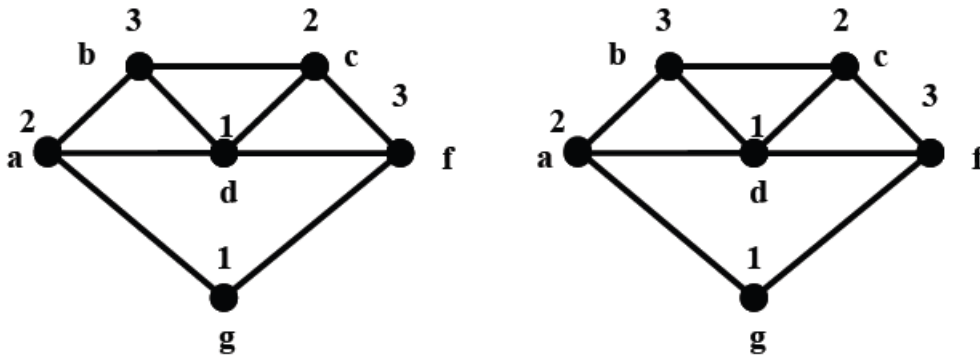
$$\text{deg}^k(v_i) = \sum_{j=1}^p a_{ij} \times \text{deg}^{k-1}(v_j), \quad i = \overline{1, p}.$$

2. «Последними – наименьшие» или ПН-упорядочение

Выбираем в исходном графе вершину с наименьшей степенью и присваиваем ей номер p . Удаляем эту вершину со всеми инцидентными ей ребрами. В полученном графе находим вершину с наименьшей степенью и присваиваем ей номер $p-1$ и т. д.

Например:

НП-упорядочение (d, a, b, c, f, g) ПН-упорядочение (d, c, f, b, a, g)



Раскраска ребер или реберная раскраска

Пусть есть $G = (V, E)$, $|V| = p$, $|E| = q$.

Реберной k -раскраской графа G называется некоторая функция ϕ , задающая отображение множества ребер графа в некоторое k -элементное множество, т.е. $\phi : E \rightarrow A = \{a_1, \dots, a_k\}$.

Если $\phi(e) = c$, то говорят, что ребро e окрашено в цвет c .

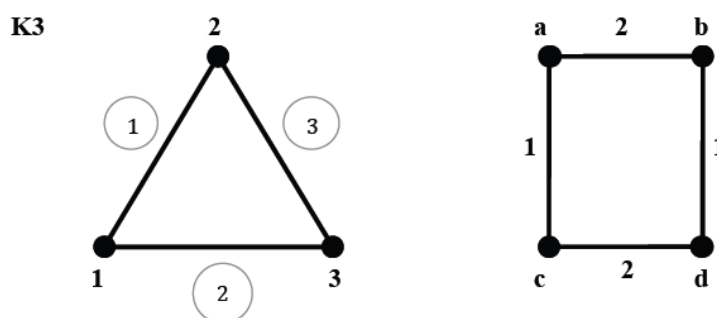
Реберная раскраска называется **правильной**, если смежные ребра окрашены в разные цвета.

Граф G называется **k -раскрашиваемым**, если существует правильная k -раскраска ребер.

Минимальное число k , при котором существует правильная реберная k -раскраска называется **реберным хроматическим числом** или **хроматическим индексом**.

Граф G называется **реберно k -хроматическим**, если хроматический индекс равен k : $\chi'(G) = k$.

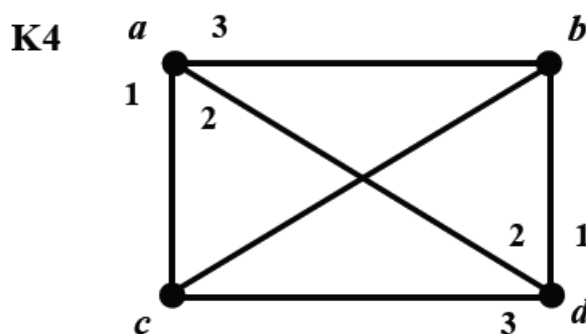
Множество ребер, окрашенных в определенный цвет, называют **реберным цветным классом**.



Хроматический индекс для полного графа с четным числом вершин равен:

$\chi'(K_{2n}) = 2n - 1$ и с нечетным числом вершин $\chi'(K_{2n+1}) = 2n + 1$.

Пример можно проиллюстрировать:



Задание к лабораторной работе

Исходные данные граф G : $G(13, \{5, 6\})$.

- 1) Планарный граф из лабораторной работы №6 обозначить $G1$ (исходный или преобразованный), а непланарный – $G2$.
- 2) Вычислить и проанализировать для планарного и непланарного графов верхние и нижние оценки хроматического числа.

- 3) Последовательно раскрасить графы G_1 и G_2 , используя алгоритм последовательной раскраски, модификации алгоритма с НП- и ПН-упорядочением вершин.
- 4) Найти хроматическое число и хроматический индекс графов G_1 и G_2 . Ответ обосновать.
- 5) Сравнить хроматическое число графов G_1 и G_2 с оценками, полученными аналитически в задании 2 и в результате применения трех алгоритмов, в задании 3. Проанализировать полученные результаты.
- 6) Привести пример графа, у которого число красок будет зависеть от порядка обхода вершин.

Контрольные вопросы

1. Вершинная раскраска неориентированных графов
2. Какая раскраска называется правильной?
3. Для каких графов могут быть применены алгоритмы раскраски?
4. Какой граф называют правильно раскрашенным?
5. Что называется хроматическим числом графа?
6. Нижние оценки хроматического числа графа.
7. Верхние оценки хроматического числа графа.
8. Определение цветного класса.
9. Сформулировать теорему Кенига.
10. Сформулировать гипотезу четырех и пяти красок.
11. Алгоритм последовательной вершинной раскраски графов.
12. Последовательные методы раскрашивания, основанные на упорядочении множества вершин. НП- и ПН-упорядочение вершин.
13. k -шаговая степень вершины и рекуррентная формула ее вычисления.
14. Реберная раскраска или раскраска ребер.
15. Правильная реберная раскраска, реберный цветной класс.
16. Определение реберного хроматического числа или хроматического индекса.
17. Хроматический индекс полных графов.

Лабораторная работа № 9

Способы задания конечных автоматов

Цель работы: изучение способов задания цифровых автоматов. Получение практических навыков в использовании алгоритмов абстрактного синтеза цифровых автоматов.

Теоретическая справка

Математической моделью конечного цифрового автомата является *абстрактный автомат*, определяемый шестью компонентами:

$$S = \{A, Z, W, \delta, \lambda, a_1\},$$

где 1) $A = \{a_1, \dots, a_m, \dots, a_M\}$ - множество состояний (алфавит состояний);

2) $Z = \{z_1, \dots, z_f, \dots, z_F\}$ - множество входных сигналов (входной алфавит);

3) $W = \{w_1, \dots, w_g, \dots, w_G\}$ - множество выходных сигналов (выходной алфавит);

4) функция переходов δ определяет правила перехода автомата из одного состояния в другое в зависимости от значений входных сигналов и состояния автомата: $\delta(a_m, z_f)$;

5) функция выходов λ определяет правила формирования выходных сигналов автомата: $\lambda(a_m, z_f)$;

6) a_1 - начальное состояние автомата ($a_1 \in A$).

Автомат имеет один вход и один выход (рис.9.1) и работает в некотором идеализированном дискретном времени, принимающим целые неотрицательные значения $t = 0, 1, 2, \dots$.

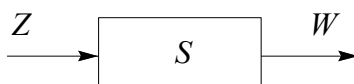


Рис.9.1. Абстрактный автомат

Если два абстрактных автомата с общим входным и выходным алфавитом индуцируют одно и то же отображение множества слов во входном алфавите во множество слов в выходном алфавите, то такие автоматы называются **эквивалентными**.

На практике наибольшее распространение получили два класса автоматов - *автоматы Мили и Мура*.

Закон функционирования **автомата Мили** задаётся уравнениями

$$\begin{aligned} a(t+1) &= \delta(a(t), z(t)); \\ w(t) &= \lambda(a(t), z(t)); \\ a(0) &= a_1, \quad t = 0, 1, 2, \dots \end{aligned} \quad (9.1)$$

Закон функционирования **автомата Мура** задаётся уравнениями

$$\begin{aligned} a(t+1) &= \delta(a(t), z(t)); \\ w(t) &= \lambda(a(t)); \\ a(0) &= a_1, \quad t = 0, 1, 2, \dots \end{aligned} \quad (9.2)$$

Как видно из уравнений (2.1) и (2.2) (их принято называть *каноническими*), эти автоматы различаются способом определения выходного сигнала. В автомате Мили функция λ определяет выходной сигнал в зависимости от состояния автомата и входного сигнала в момент времени t , а в автомате Мура накладываются ограничения на функцию λ , заключающиеся в том, что выходной сигнал зависит только от состояния автомата и не зависит от значения входных сигналов.

Важным отличием функционирования этих автоматов является то, что выходные сигналы автомата Мура отстают на один такт от выходных сигналов автомата Мили, эквивалентного ему.

Стандартные автоматные языки

Для задания автоматов общего типа используются *стандартные или автоматные языки*, задающие функции переходов и выходов в явном виде. К ним относятся *таблицы, графы, матрицы переходов и выходов* и их

аналитическая интерпретация. Для того, чтобы задать автомат, необходимо описать все компоненты вектора $S = (A, Z, W, \delta, \lambda, a_1)$.

При табличном способе задания автомат Мили описывается с помощью двух таблиц: *таблицы переходов* и *таблицы выходов*. **Таблица переходов** задает функцию δ (табл.9.1.), **таблица выходов** - функцию λ (табл.9.2.). Каждому столбцу табл.9.1 и 9.2 поставлено в соответствие одно состояние из множества A , каждой строке - один входной сигнал из множества Z . На пересечении столбца a_m и строки z_f в табл.9.1 записывается состояние a_s , в которое должен перейти автомат из состояния a_m под действием входного сигнала z_f , т.е. $a_s = \delta(a_m, z_f)$. На пересечении столбца a_m и строки z_f в табл.9.2 записывается выходной сигнал w_g , выдаваемый автоматом в состоянии a_m при поступлении на его вход сигнала z_f , т.е. $w_g = \lambda(a_m, z_f)$

Таблица 9.1

	a_1	a_2	a_3	a_4
z_1	a_2	a_2	a_1	a_1
z_2	a_4	a_3	a_4	A_3

Таблица 9.2

	a_1	a_2	a_3	a_4
z_1	w_1	w_1	w_2	w_4
z_2	w_5	w_3	w_4	w_5

Для указанных таблиц $A = \{a_1, a_2, a_3, a_4\}$; $Z = \{z_1, z_2\}$; $W = \{w_1, w_2, w_3, w_4, w_5\}$.

Автомат Мили может быть задан также одной совмещенной таблицей переходов и выходов (табл.9.3), в которой каждый элемент a_s/w_g , записанный на пересечении столбца a_m и строки z_f , определяется следующим образом:

$$a_s = \delta(a_m, z_f); \quad w_g = \lambda(a_m, z_f).$$

Таблица 9.3

	a_1	a_2	a_3	a_4
z_1	a_2/w_1	a_2/w_1	a_1/w_2	a_1/w_4
z_2	a_4/w_5	a_3/w_3	a_4/w_4	a_3/w_5

Таблица 9.4

	w_3	w_2	w_3	w_1
	a_1	a_2	a_3	a_4
z_1	a_1	a_3	a_1	a_4
z_2	a_2	a_4	a_4	a_1

Автомат Мура задается одной отмеченной таблицей переходов (табл.9.4), в которой каждому столбцу приписаны не только состояния a_m , но еще и

выходной сигнал w_g , соответствующий этому состоянию, где $w_g = \lambda(a_m)$. Для табл. 9.4 $A = \{a_1, a_2, a_3, a_4\}$; $Z = \{z_1, z_2\}$; $W = \{w_1, w_2, w_3\}$.

Автомат называется *неполностью определенным* или *частичным*, если либо функция δ определена не на всех парах $(a_m, z_f) \in A \times Z$, либо функция λ определена не на всех указанных парах в случае автомата Мили и на множестве не всех внутренних состояний для автомата Мура. Для частичных автоматов Мили и Мура в рассмотренных таблицах на месте неопределенных состояний и выходных сигналов ставится прочерк.

Граф автомата - это ориентированный граф, вершины которого соответствуют состояниям, а дуги - переходам между ними. Дуга, направленная из вершины a_m в вершину a_s , задает переход в автомате из состояния a_m в состояние a_s . В начале этой дуги записывается входной сигнал $z_f \in Z$, вызывающий данный переход: $a_s = \delta(a_m, z_f)$. Для графа автомата Мили выходной сигнал $w_g \in W$, формируемый на переходе, записывается в конце дуги, а для автомата Мура - рядом с вершиной a_m , отмеченной символом состояния a_m , в котором он формируется. Если переход в автомате из состояния a_m в состояние a_s производится под действием нескольких входных сигналов, то дуге графа, направленной из a_m в a_s , приписываются все эти входные и соответствующие выходные сигналы. Графы автоматов Мили и Мура, построенные по табл.9.3 и 9.4 приведены соответственно на рис.9.2. а, б.

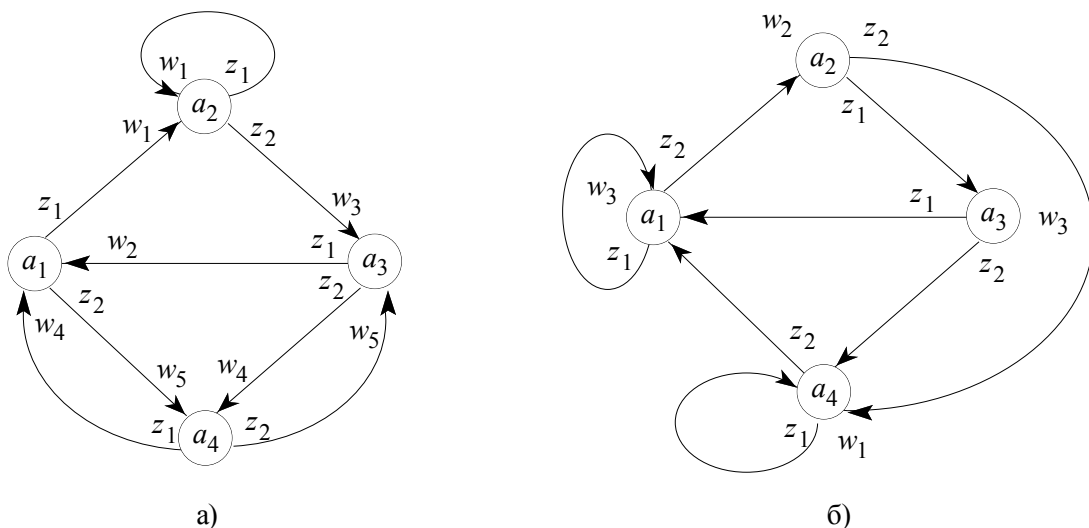


Рис.9.2. Граф автомата Мили (а) и автомата Мура (б)

Применительно к графу условия однозначности и полной определенности будут заключены в следующем:

- не существует двух ребер с одинаковыми входными пометками, выходящих из одной и той же вершины;
- для всякой вершины a_m и для всякого входного сигнала z_f имеется ребро, помеченное символом z_f , которое выходит из a_m .

При задании графов с большим числом состояний и переходов наглядность теряется, поэтому оказывается предпочтительным задавать этот граф в виде списка - *таблицы переходов*.

Прямая таблица переходов - таблица в которой последовательно перечисляются все переходы сначала из первого состояния, затем из второго и т.д. Табл.9.5 - прямая таблица переходов автомата Мили, построенная по графу, приведенному на рис.9.2.,а. В ряде случаев оказывается удобным пользоваться **обратной таблицей переходов**, в которой столбцы обозначены точно так же, но сначала записываются все переходы в первое состояние, затем во второе и т.д. Табл.9.6 - обратная таблица переходов автомата Мура, построенная по графу, приведенному на рис.9.2.,б. Как и граф, таблицы переходов должны удовлетворять условиям однозначности и полноты переходов.

В некоторых случаях для задания автомата используются **матрицы переходов и выходов**, строки и столбцы которой отмечены символами состояний. Если существует переход из a_m под действием z_f в a_s с выдачей w_g , то на пересечении строки a_m и столбца a_s записывается пара $z_f w_g$. Ясно, что не всякая матрица задает автомат. Как граф и таблица переходов и выходов, она должна удовлетворять условиям однозначности и полноты переходов.

Системы канонических уравнений (СКУ) и системы выходных функций (СВФ) являются аналитической интерпретацией таблиц переходов и выходов или графов автоматов. СКУ определяет функции переходов автомата, а СВФ - функции выходов. Каждое состояние автомата интерпретируется как событие, соответствующее множеству переходов в это состояние:

$$a_s(t+1) = \bigvee_{f,m} z_f(t) \& a_m(t). \quad (9.5)$$

Таблица.9.5

$a_m(t)$	$z_f(t)$	$a_s(t+1)$	$w_g(t)$
a_1	z_1	a_2	w_1
	z_2	a_4	w_5
a_2	z_1	a_2	w_1
	z_2	a_3	w_3
a_3	z_1	a_1	w_2
	z_2	a_4	w_4
a_4	z_1	a_1	w_4
	z_2	a_3	w_5

Таблица 9.6

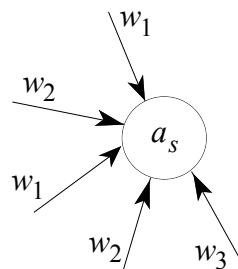
$a_m(t)$	$z_f(t)$	$a_s(t+1), w_g(t+1)$
a_1	z_1	a_1, w_3
a_3	z_1	
a_4	z_2	
a_1	z_2	a_2, w_2
a_2	z_1	a_3, w_3
a_2	z_2	a_4, w_1
a_3	z_2	
a_4	z_1	

Связь между автоматами Мура и Мили

Для любого автомата Мили можно построить эквивалентный ему автомат Мура и, наоборот, для любого автомата Мура можно построить эквивалентный ему автомат Мили.

Рассмотрим преобразование автомата Мили в автомат Мура. Пусть задан автомат Мили $S = (A, Z, W, \delta, \lambda, a_1)$, построим автомат Мура $S' = (A', Z', W', \delta', \lambda', a_1')$, у которого $Z' = Z$; $W' = W$.

Для определения множества A' необходимо выполнить расщепление состояний автомата Мили, исходя из следующего. Если автомат Мили при переходе в некоторое состояние a_s может выдавать в разные моменты времени один из k выходных сигналов из алфавита W , то такое состояние a_s должно быть расщеплено на k состояний. То есть число элементов в множестве A_s равно числу различных выходных сигналов на дугах автомата S , входящих в состояние a_s : $A_s = \{(a_s, w_1), (a_s, w_2), (a_s, w_3), \dots, (a_s, w_k)\}$ (рис.9.3).



$$A_s = \{(a_s, w_1), (a_s, w_2), (a_s, w_3)\}$$

Рис.9.3. Построение множества A_s

Множество состояний автомата S' получим как объединение множеств A_s

$$A' = \bigcup_{S=1}^M A_s,$$

где M - число состояний в автомате Мили S .

Такое расщепление состояний автомата Мили необходимо потому, что все состояния эквивалентного ему автомата Мура должны быть отмечены только одним выходным сигналом из алфавита W .

Функцию переходов δ' и выходов λ' определим следующим образом. Если в автомате Мили S был переход $\delta(a_m, z_f) = a_s$ и при этом выдавался выходной сигнал $\lambda(a_m, z_f) = w_g$, то в автомате Мура S' (рис.9.4) будет переход из множества состояний A_m , порождаемых a_m , в состояние (a_s, w_g) под действием того же входного сигнала z_f .

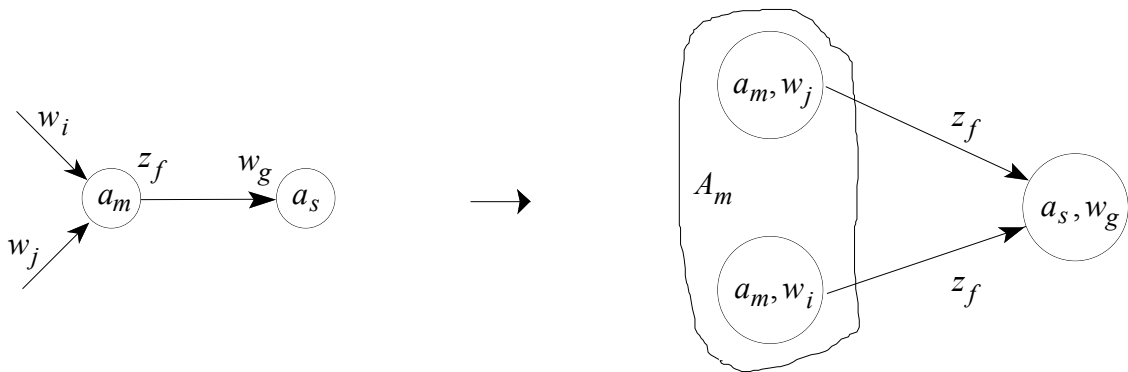


Рис.9.4. Иллюстрация перехода от автомата Мили к автомату Мура

В качестве начального состояния автомата Мура a'_1 можно взять любое из состояний множества A_1 , порождаемого начальным состоянием a_1 .

Пример. Преобразовать автомат Мили, изображенный на рис.9.2,а, в автомат Мура.

Решение. Для автомата Мура $Z' = Z = \{z_1, z_2\}$; $W' = W = \{w_1, w_2, w_3, w_4, w_5\}$.

Для построения множества A' нужно найти множества пар, порождаемых каждым состоянием автомата Мили S . Каждую пару обозначим символами

b_1, b_2, \dots :

$$\begin{aligned} A_1 &= \{(a_1, w_2), (a_1, w_4)\} = \{b_1, b_2\}; & A_2 &= \{(a_2, w_1)\} = \{b_3\}; \\ A_3 &= \{(a_3, w_3), (a_3, w_5)\} = \{b_4, b_5\}; & A_4 &= \{(a_4, w_4), (a_4, w_5)\} = \{b_6, b_7\}. \end{aligned}$$

$$A' = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}.$$

Для определения функции λ' с каждым состоянием вида (a_s, w_g) , представляющим собой пару, отождествим выходной сигнал, являющийся вторым элементом этой пары:

$$\begin{aligned} \lambda'(b_1) &= w_2; & \lambda'(b_2) &= \lambda'(b_6) = w_4; & \lambda'(b_3) &= w_1; \\ \lambda'(b_4) &= w_3; & \lambda'(b_5) &= \lambda'(b_7) = w_5. \end{aligned}$$

Функция δ' строится следующим образом. Так как в автомате Мили S есть переход из состояния a_1 под действием сигнала z_1 в состояние a_2 с выдачей w_1 , то из множества состояний $A_1 = \{b_1, b_2\}$, порождаемых a_1 в автомате S' должен быть переход в состояние $(a_2, w_1) = b_3$ под действием сигнала z_1 . Аналогично из состояний множества $A_1 = \{b_1, b_2\}$ должен быть переход в состояние $(a_4, w_5) = b_7$ под действием сигнала z_2 и т.д. В качестве начального состояния можно выбрать одно из состояний, порождаемых a_1 , например b_1 . На рис.9.5 приведен граф автомата Мура S' , эквивалентного автомату Мили S .

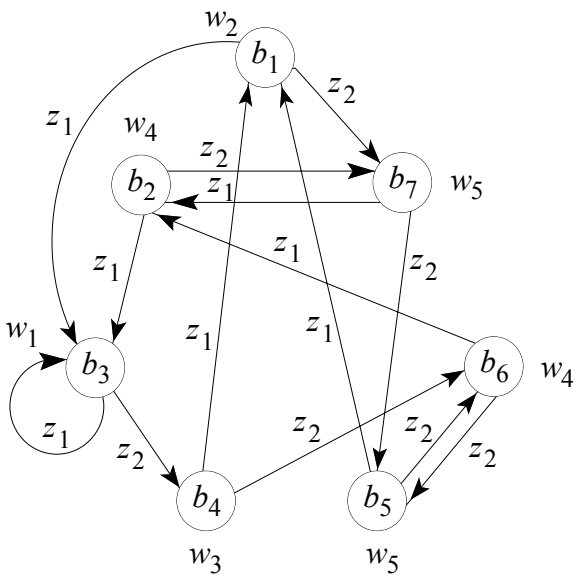


Таблица 9.7

	w_2	w_4	w_1	w_3	w_5	w_4	w_5
	b_1	b_2	b_3	b_4	b_5	b_6	b_7
z_1	b_3	b_3	b_3	b_1	b_1	b_2	b_2
z_2	b_7	b_7	b_4	b_6	b_6	b_5	b_5

Рис.9.5. Граф автомата Мура, эквивалентного автомату Мили (рис. 9.2,а)

Задание к лабораторной работе

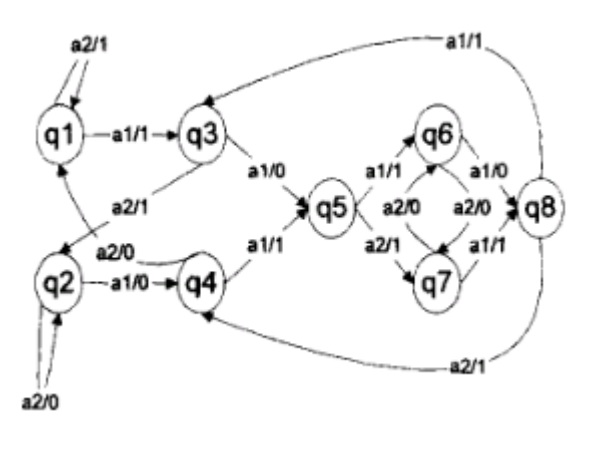
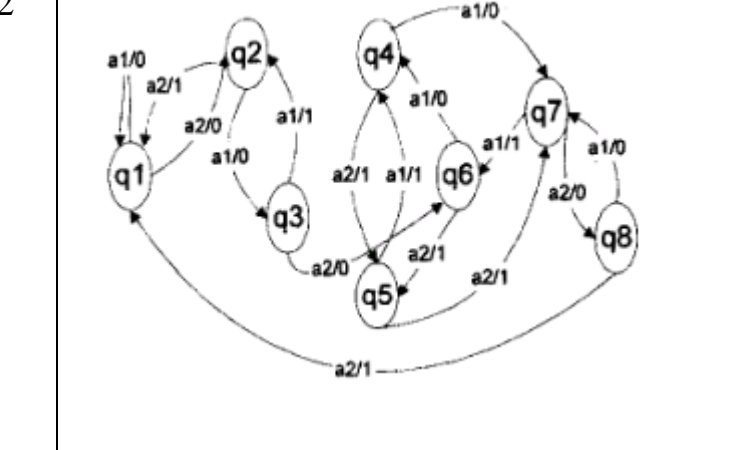
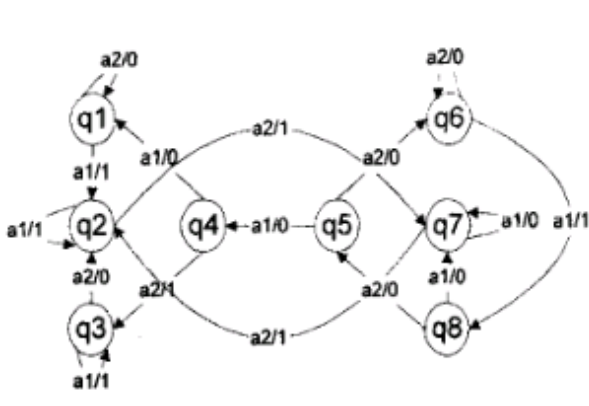
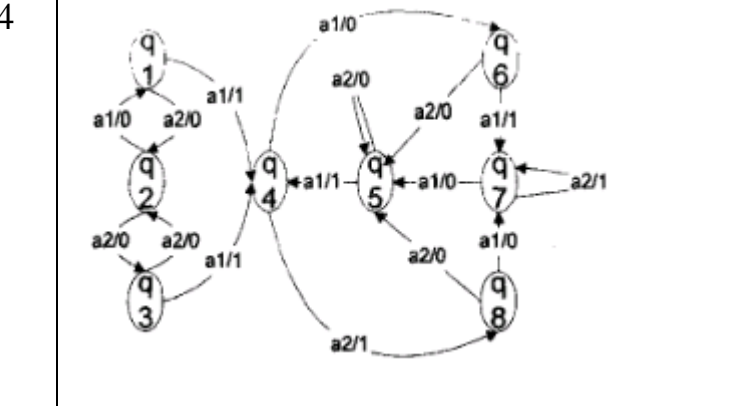
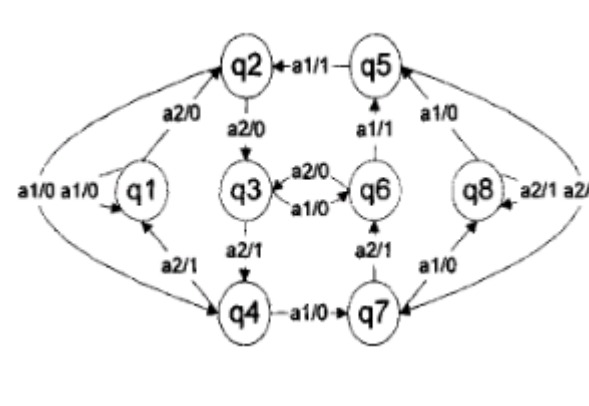
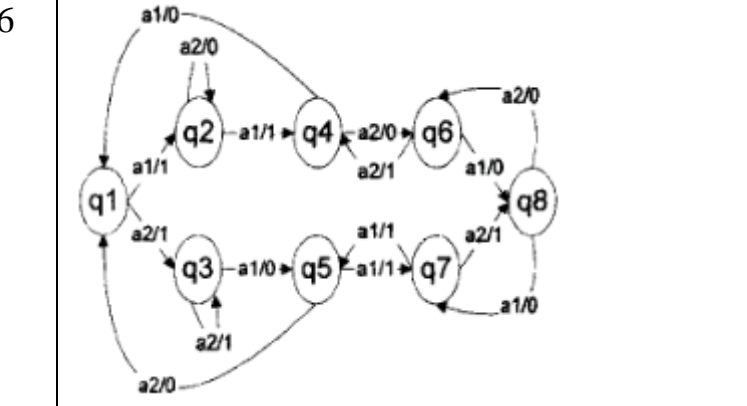
Автомат Мили задан графом. Варианты заданий представлены в таблице 9.8.

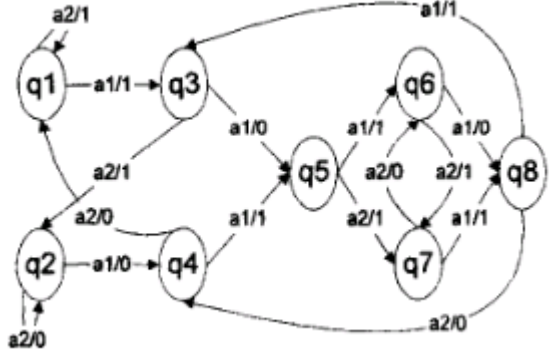

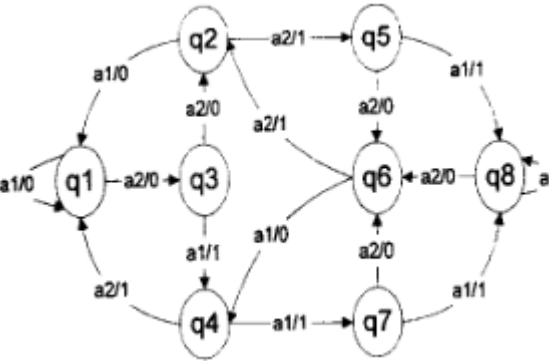
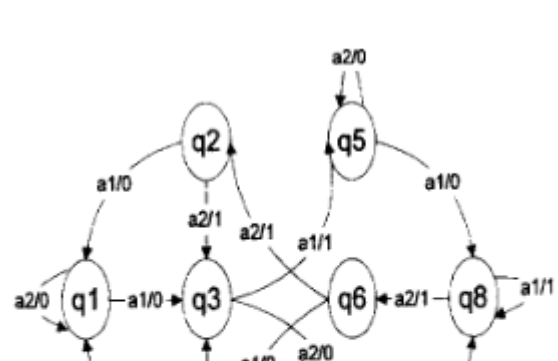
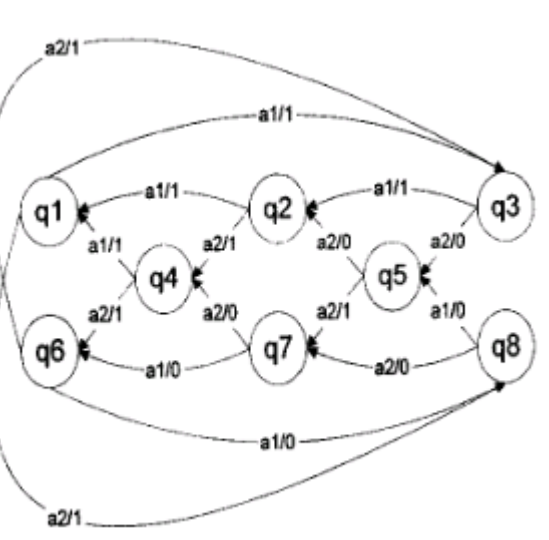
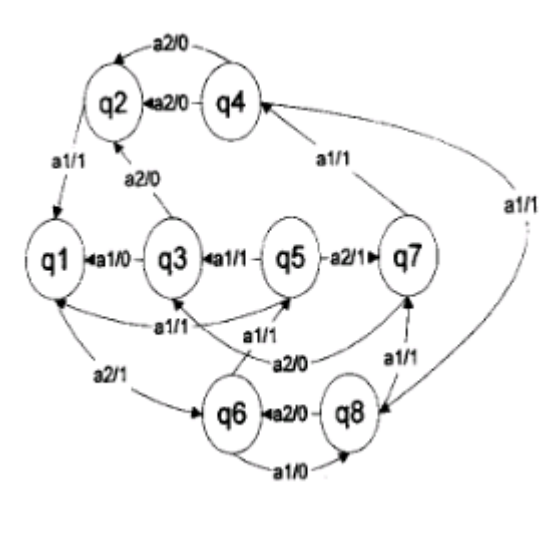
Необходимо выполнить следующие действия:

- построить совмещенную таблицу переходов/выходов;

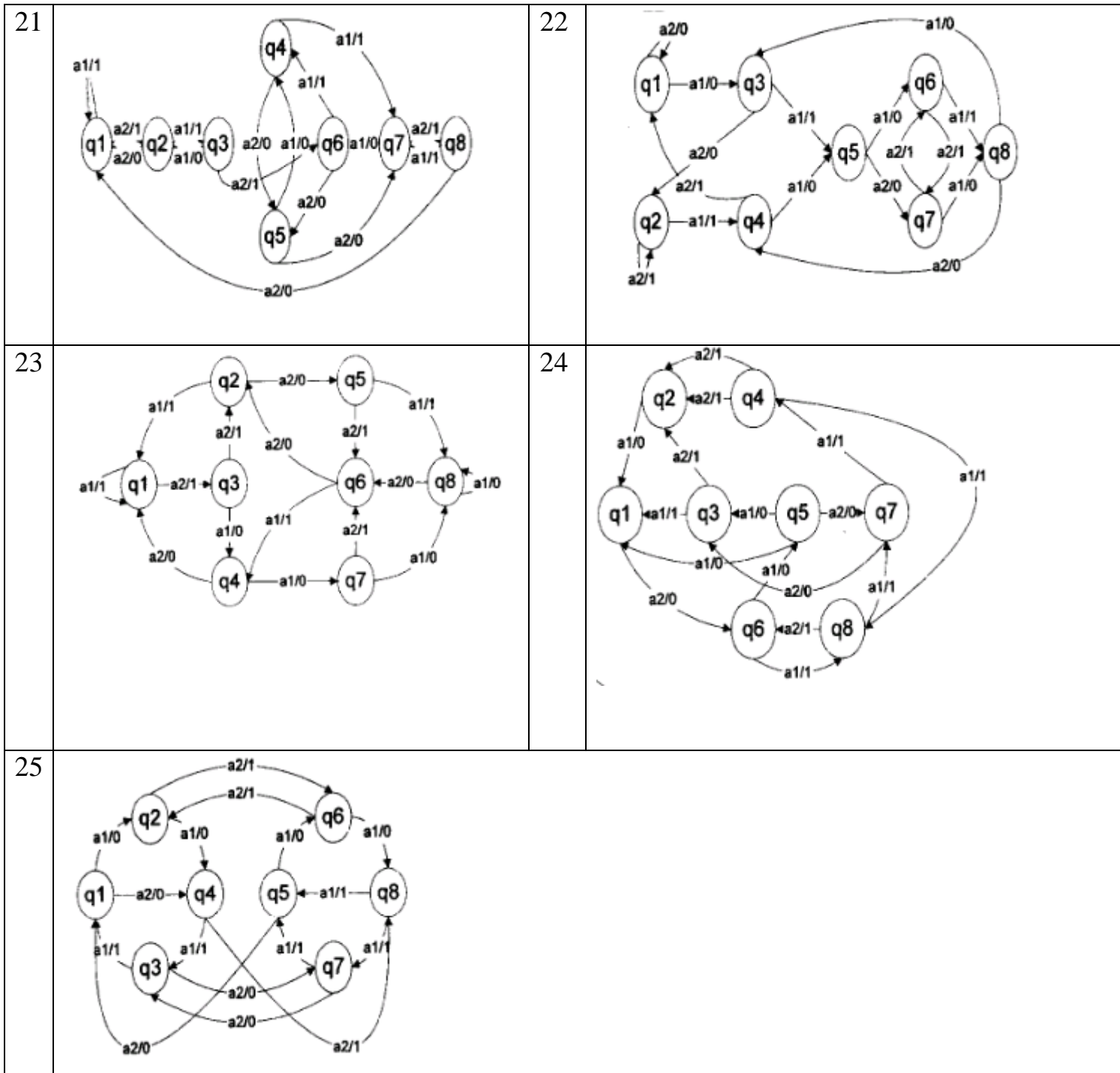
- определить последовательность состояний и выходное слово для произвольного входного слова из 10 символов;
- выполнить графически переход от автомата Мили к автомату Мура.

Таблица 9.8 Варианты заданий

1		2	
3		4	
5		6	
7		8	

		
9		
11		

<p>13</p>	<p>Diagram 13: A state transition diagram with 8 states (q1 to q8). Transitions are labeled with pairs (action, output). Key transitions include: q1 to q2 (a1/0), q2 to q1 (a2/0), q2 to q4 (a1/1), q4 to q2 (a2/1), q3 to q1 (a1/0), q1 to q3 (a2/1), q3 to q5 (a1/0), q5 to q3 (a2/1), q4 to q5 (a1/0), q5 to q4 (a2/1), q6 to q1 (a2/0), q1 to q6 (a1/0), q6 to q8 (a1/1), q8 to q6 (a2/0), q7 to q5 (a1/0), q5 to q7 (a2/1), q7 to q8 (a1/1), q8 to q7 (a2/0).</p>	<p>14</p>	<p>Diagram 14: A state transition diagram with 8 states (q1 to q8). Transitions include: q1 to q2 (a1/1), q2 to q1 (a2/1), q1 to q3 (a1/1), q3 to q1 (a2/0), q2 to q4 (a1/1), q4 to q2 (a2/0), q3 to q5 (a1/0), q5 to q3 (a2/1), q4 to q6 (a1/0), q6 to q4 (a2/1), q5 to q7 (a1/0), q7 to q5 (a2/0), q6 to q8 (a1/0), q8 to q6 (a2/1), q7 to q8 (a1/0), q8 to q7 (a2/0).</p>
<p>15</p>	<p>Diagram 15: A state transition diagram with 8 states (q1 to q8). Transitions include: q1 to q2 (a1/1), q2 to q1 (a2/0), q1 to q3 (a2/1), q3 to q1 (a1/1), q2 to q4 (a1/0), q4 to q2 (a2/1), q3 to q5 (a2/1), q5 to q3 (a1/0), q4 to q6 (a1/1), q6 to q4 (a2/0), q5 to q7 (a2/0), q7 to q5 (a1/0), q6 to q8 (a1/0), q8 to q6 (a2/1), q7 to q8 (a1/1), q8 to q7 (a2/0).</p>	<p>16</p>	<p>Diagram 16: A state transition diagram with 8 states (q1 to q8). Transitions include: q1 to q3 (a1/0), q3 to q1 (a2/0), q1 to q2 (a2/1), q2 to q1 (a1/1), q3 to q5 (a1/1), q5 to q3 (a2/0), q2 to q4 (a1/0), q4 to q2 (a2/1), q5 to q6 (a1/0), q6 to q5 (a2/1), q4 to q7 (a1/0), q7 to q4 (a2/0), q5 to q8 (a1/0), q8 to q5 (a2/1), q6 to q8 (a1/1), q8 to q6 (a2/0).</p>
<p>17</p>	<p>Diagram 17: A state transition diagram with 8 states (q1 to q8). Transitions include: q1 to q2 (a1/1), q2 to q1 (a2/0), q1 to q3 (a2/1), q3 to q1 (a1/0), q2 to q4 (a2/1), q4 to q2 (a1/1), q3 to q5 (a2/1), q5 to q3 (a1/0), q4 to q6 (a1/1), q6 to q4 (a2/0), q5 to q7 (a1/1), q7 to q5 (a2/1), q6 to q8 (a1/0), q8 to q6 (a2/1), q7 to q8 (a1/1), q8 to q7 (a2/0).</p>	<p>18</p>	<p>Diagram 18: A state transition diagram with 8 states (q1 to q8). Transitions include: q1 to q2 (a1/0), q2 to q1 (a2/1), q1 to q3 (a1/1), q3 to q1 (a2/0), q2 to q4 (a1/1), q4 to q2 (a2/1), q3 to q5 (a1/0), q5 to q3 (a2/1), q4 to q6 (a1/1), q6 to q4 (a2/0), q5 to q7 (a1/1), q7 to q5 (a2/1), q6 to q8 (a1/0), q8 to q6 (a2/1), q7 to q8 (a1/1), q8 to q7 (a2/0).</p>
<p>19</p>	<p>Diagram 19: A state transition diagram with 8 states (q1 to q8). Transitions include: q1 to q2 (a1/1), q2 to q1 (a2/1), q1 to q3 (a1/0), q3 to q1 (a2/0), q2 to q4 (a1/0), q4 to q2 (a2/1), q3 to q5 (a1/1), q5 to q3 (a2/0), q4 to q6 (a2/1), q6 to q4 (a1/0), q5 to q7 (a1/0), q7 to q5 (a2/1), q6 to q8 (a1/1), q8 to q6 (a2/0), q7 to q8 (a1/1), q8 to q7 (a2/0).</p>	<p>20</p>	<p>Diagram 20: A state transition diagram with 8 states (q1 to q8). Transitions include: q1 to q2 (a2/0), q2 to q1 (a1/0), q1 to q3 (a1/0), q3 to q1 (a2/1), q2 to q4 (a2/1), q4 to q2 (a1/1), q3 to q5 (a2/1), q5 to q3 (a1/1), q4 to q6 (a2/0), q6 to q4 (a1/1), q5 to q7 (a1/0), q7 to q5 (a2/0), q6 to q8 (a2/1), q8 to q6 (a1/1), q7 to q8 (a2/0), q8 to q7 (a1/1).</p>



Контрольные вопросы

1. Дать определение абстрактного автомата.
2. Что значит задать конечный автомат ?
3. Перечислить способы задания автоматов.
4. Описать закон функционирования автомата Мили.
5. Описать закон функционирования автомата Мура.
6. Чем отличается автомат Мили от автомата Мура.

7. Чем отличается автомат Мили от автомата Мура ?
8. Привести пример графического способа задания автомата Мили.
9. Привести пример графического способа задания автомата Мура.
10. Что называется таблицей переходов автомата?
11. Привести пример таблицы выходов автомата Мили.
12. Привести пример таблицы выходов автомата Мура.
13. Что называется отмеченной таблицей переходов и для какого автомата она задается?
14. Проиллюстрировать графически переход от автомата Мура к автомату Мили.
15. Проиллюстрировать графически переход от автомата Мили к автомату Мура

Лабораторная работа № 10

Минимизация конечных автоматов

Цель работы: изучение методов минимизации автоматов, построение минимальных автоматов.

Теоретическая справка

Постановка задачи минимизации

Задача минимизации заключается в отыскании автомата с минимальным числом состояний среди всех автоматов, реализующих заданные условия работы. Под заданными условиями работы понимаются входной и выходной алфавиты автоматов. Общий подход во всех методах минимизации автоматов общего типа заключается в последовательном уменьшении заданного числа состояний до тех пор, пока автомат реализует заданные условия работы.

Метод Ауфенкампа и Хона

Основная идея этого метода состоит в разбиении всех состояний исходного абстрактного автомата на попарно непересекающиеся классы эквивалентных состояний и замене каждого класса эквивалентности одним состоянием. Получающийся в результате минимизации автомат имеет столько состояний, на сколько классов эквивалентности разбиваются состояния исходного автомата.

Состояния a_m и a_s называются **эквивалентными** ($a_m \sim a_s$), если $\lambda(a_m, \xi) = \lambda(a_s, \xi)$ для всевозможных входных слов ξ . Состояния a_m и a_s называются **k -эквивалентными** ($a_m \overset{k}{\sim} a_s$), если $\lambda(a_m, \xi) = \lambda(a_s, \xi)$ для всевозможных входных слов длины k .

Минимизация автомата Мили

Алгоритм минимизации автомата Мили $S = \{A, Z, W, \delta, \lambda, a_1\}$ состоит из следующих шагов:

1. Находятся последовательные разбиения $\pi_1, \pi_2, \dots, \pi_k, \pi_{k+1}$ множества A на классы одно-, двух-, \dots , K -, $K+1$ - эквивалентных состояний до тех пор, пока на каком-то $(K+1)$ шаге не окажется, что $\pi_k = \pi_{k+1}$.

Одноэквивалентными будут состояния с одинаковыми столбцами в таблице выходов. Состояния будут *двухэквивалентными*, если они одноэквивалентны и под действием одинаковых входных сигналов попадают в одинаковые одноэквивалентные классы.

2. В каждом классе эквивалентности разбиения π выбирается по одному состоянию, в результате чего получается множество A' состояний минимального автомата $S' = \{A', Z, W, \delta', \lambda', \}$, эквивалентного автомату S .

3. Для определения функции переходов δ' и функции выходов λ' автомата S' в таблице переходов и выходов вычёркиваются столбцы, соответствующие не вошедшим в A' состояниям. В оставшихся столбцах не вошедшие в множество A состояния заменяются на эквивалентные.

4. В качестве начального состояния a_1' выбирается состояние, эквивалентное состоянию a_1 . В частности, удобно за a_1' принимать само состояние a_1 .

Пример 1. Минимизировать полностью определённый автомат Мили S_1 , заданный таблицами переходов и выходов (табл. 10.1 и 10.2).

Таблица 10.1

	a_1	a_2	a_3	a_4	a_5	a_6			a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_3	a_4	a_3	a_4	a_5	a_6		z_1	w_1	w_1	w_1	w_1	w_1	w_1
z_2	a_5	a_6	a_5	a_6	a_1	a_2		z_2	w_1	w_1	w_2	w_2	w_1	w_1

Таблица 10.2

Решение.

1. По таблице выходов (табл. 10.2) находим разбиение π_1 на классы одноэквивалентных состояний, объединяя одинаковые столбцы:

$$\pi_1 = \{B_1, B_2\} = \{\{a_1, a_2, a_5, a_6\}, \{a_3, a_4\}\}.$$

Для сокращения числа скобок будем использовать надчёркивания, а элементы множества под чертой разделять точками:

$$\pi_1 = \{B_1, B_2\} = \{\overline{1.2.5.6}, \overline{3.4}\}.$$

Строим таблицу π_1 (табл. 10.3), заменяя состояния в таблице переходов исходного автомата (табл. 10.1) соответствующими классами одноэквивалентности.

Таблица 10.3

	B_1				B_2	
	a_1	a_2	a_5	a_6	a_3	a_4
z_1	B_2	B_2	B_1	B_1	B_2	B_2
z_2	B_1	B_1	B_1	B_1	B_1	B_1

По табл. 3 получим разбиение π_2 на классы 2-эквивалентных состояний (табл. 10.4):

$$\pi_2 = \{C_1, C_2, C_3\} = \{\overline{1.2}, \overline{5.6}, \overline{3.4}\}.$$

Таблица 10.4

	C_1		C_2		C_3	
	a_1	a_2	a_5	a_6	a_3	a_4
z_1	C_3	C_3	C_2	C_2	C_3	C_3
z_2	C_2	C_2	C_1	C_1	C_2	C_2

Разбиение π_3 получаем аналогично (табл. 10.4):

$$\pi_3 = \{D_1, D_2, D_3\} = \{\overline{1.2}, \overline{5.6}, \overline{3.4}\}.$$

Оно полностью совпадает с π_2 . Процедура завершена. Разбиение $\pi_3 = \pi_2 = \pi$ есть разбиение множества состояний автомата Мили S_1 на классы эквивалентных между собой состояний.

2. Из каждого класса эквивалентности произвольно выбираем по одному состоянию: $A' = \{a_1, a_3, a_6\}$.

3. Строим таблицы переходов и выходов минимального автомата S'_1 (табл. 10.5 и 10.6).

Таблица 5

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_3	a_4	a_3	a_4	a_5	a_6
z_1	a_5 a_6	a_6	a_5 a_6	a_6	a_1	a_2 a_1

Таблица 6

	a_1	a_3	a_6
z_1	a_3	a_3	a_6
z_2	a_6	a_6	a_1

Таблица 7

	a_1	a_2	a_6
z_1	w_1	w_1	w_1
z_1	w_1	w_2	w_1

Минимизация автомата Мура

При минимизации полностью определённых автоматов Мура вводится понятие 0-эквивалентности состояний и разбиение множества состояний на 0-эквивалентные классы. 0-эквивалентными являются одинаково отмеченные состояния. Если два состояния автомата Мура 0-эквивалентны и под действием одинаковых входных сигналов попадают в 0-эквивалентные состояния, то они называются 1-эквивалентными. Все дальнейшие классы эквивалентности для автомата Мура определяются аналогично рассмотренному выше для автомата Мили.

Пример 2. Минимизировать полностью определённый автомат Мура S_2 , заданный отмеченной таблицей переходов (табл. 10.8)

Таблица 10.8

	w_1	w_1	w_3	w_3	w_3	w_2	w_3	w_1	w_2	w_2	w_2	w_2
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
z_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7	a_3	a_{10}	a_7	a_1	a_5	a_2
z_2	a_5	a_7	a_6	a_{11}	a_9	a_{11}	a_6	a_4	a_6	a_8	a_9	a_8

Решение.

1. По табл. 10.8 находим разбиение π_0 на классы 0-эквивалентных состояний, отыскивая одинаково отмеченные состояния:

$$\pi_0 = \{A_1, A_2, A_3\} = \{ \overline{1.2.8}, \overline{6.9.10.11.12}, \overline{3.4.5.7} \} .$$

Строим таблицу π_0 (табл. 10.9), заменяя состояния в таблице переходов (табл. 8) соответствующими классами 0-эквивалентности.

Таблица 10.9

	A_1			A_2					A_3			
	a_1	a_2	a_8	a_6	a_9	a_{10}	a_{11}	a_{12}	a_3	a_4	a_5	a_7
z_1	A_2	A_2	A_2	A_3	A_3	A_1	A_3	A_1	A_3	A_3	A_3	A_3
z_2	A_3	A_3	A_3	A_2	A_2	A_1	A_2	A_1	A_2	A_2	A_2	A_2

По табл. 10.9 получим разбиение π_1 на классы 1-эквивалентных состояний (табл. 10.10):

$$\pi_1 = \{B_1, B_2, B_3, B_4\} = \{\overline{1.2.8}, \overline{6.9.11}, \overline{10.12}, \overline{3.4.5.7}\}.$$

Таблица 10.10

	$B1$			$B2$			$B3$		$B4$			
	$a1$	$a2$	$a8$	$a6$	$a9$	$a11$	$a10$	$a12$	$a3$	$a4$	$a5$	$A7$
$z1$	$B3$	$B3$	$B3$	$B4$	$B4$	$B4$	$B1$	$B1$	$B4$	$B4$	$B4$	$B4$
$z2$	$B4$	$B4$	$B4$	$B2$	$B2$	$B2$	$B1$	$B1$	$B2$	$B2$	$B2$	$B2$

Из табл. 10.10 видно, что $\pi_2 = \pi_1 = \pi$. То есть поиск классов эквивалентности завершён.

2. Из каждого класса эквивалентности произвольно выбираем по одному элементу: $A' = \{a_1, a_6, a_{10}, a_3\}$.

3. Строим отмеченную таблицу переходов минимального автомата S_2 (табл. 10.11).

Таблица 10.11

	w_1	w_3	w_2	w_2
	a_3	a_1	a_6	a_{10}
z_1	a_{10}	a_3	a_3	a_1
z_2	a_3	a_6	a_6	a_1

Задание к лабораторной работе

Выполнить минимизацию автомата Мили по алгоритму Ауфенкампа и Хона.

Преобразовать полученный минимальны автомат Мили в автомат Мура графическим способом. По полученному графу построить таблицу переходов автомата Мура.

Преобразовать полученный автомат Мура в автомат Мили графическим способом. По полученному графу построить таблицу переходов/выходов автомата Мили. Сравнить полученный автомат Мили с минимальным автоматом Мура по числу состояний, функциям выходов/переходов.

Контрольные вопросы

1. Какие состояния автомата называют эквивалентными?
2. Как строится таблица переходов автомата Мили эквивалентного автомату Мура.
3. Как определяется множество состояний автомата Мура эквивалентного автомату Мили.
4. В чем заключается задача минимизации внутренних состояний автомата?
5. Какие состояния называются эквивалентными, К-эквивалентными ?
6. Приведите основные этапы минимизации полностью определенных автоматов методом Ауфенкампа и Хона .

Литература

1. Лекции по теории графов /под ред. Емеличева Е.А./ – М.: Наука, 1990. – 384с.
2. Кристофидес Н. Теория графов : алгоритмический подход. – М.: Мир, 1978. – 432с.
3. Оре О. Теория графов. М.: Наука, 1980. – 336с.
4. Харари Ф. Теория графов. – М.: Мир, 1973. – 300с.
5. Свами М., Тхуласираман К. Графы, сети и алгоритмы. – М.: Мир, 1988. – 455с.
6. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженеров. – М.: Мир, 1980.
7. Кук Д., Бейз Г. Компьютерная математика . – М.: Наука, 1990.
8. Столл Р. Множества.Логика.Аксиоматические теории. – М.: Просвещение, 1968.
9. Поспелов Д.А. Логические методы анализа и синтеза схем. – М.: Энергия,1974. – 268с.
- 10.Шоломов Л.А. Основы теории дискретных логических вычислительных устройств. – М.: Наука,1980.
- 11.Андерсон Дж. Дискретная математика и комбинаторика.– М.: Мир, 2001. – 960с.
- 12.Новиков Ф.А. Дискретная математика для программистов. – СПб: Питер, 2000. –304с.
- 13.Уилсон Р. Введение в теорию графов. – М.: Мир,1977. – 202с.
- 14.Евстигнеев В.А. Применение теории графов в программировании. – М.: Наука, 1985. –352.

Приложение А

Алгоритм генерации варианта

$GV (p, X) : A[1:p, 1:p]$, где

p - количество вершин в графе;

X - параметр генерации (множество целых);

A - матрица смежности неориентированного графа.

$S = \langle \text{фамилия} \rangle \langle \text{имя} \rangle \langle \text{отчество} \rangle$

$n (c)$ - функция - номер буквы в алфавите (1..32)

1. Вычеркнуть из S все повторные вхождения букв.

2. Построить $Y = \| y_{ij} \|, i, j = 1..p,$

$$y_{ij} = | n (S_i) - n (S_j) |.$$

3. Построить $A = \| a_{ij} \|, i, j = 1..p,$

$$a_{ij} = \begin{cases} 1, \text{если } \exists x \in X \mid y_{ij} \bmod x = 0 \\ 0, i = j \\ 0, \text{иначе} \end{cases}$$

4. Для каждой изолированной (доминирующей) вершины добавить (удалить) одно ребро. Добавляемое (удаляемое) ребро связывает текущую вершину со следующей (по номеру). Для последней вершины следующая - первая.

Пример реализации $GV (7, (2,3))$.

1. Строка $S = \text{СИДОРОВИВАНПЕТРОВИЧ}$.

После вычеркивания повторных вхождений букв

$S = \text{СИДОРВАНПЕТЧ}$.

Таблица для функции $n (S)$

А -	Д -	З -	Л -	П -	У -	Ч -	Ь -
1	5	9	13	17	21	25	29
Б -	Е -	И -	М -	Р -	Ф -	Ш -	Э -
2	6	10	14	18	22	26	30
В -	Ё -	Й -	Н -	С -	Х -	Щ -	Ю -
3	7	11	15	19	23	27	31
Г -	Ж -	К -	О -	Т -	Ц -	Ы -	Я -
4	8	12	16	20	24	28	32

S С И Д О Р В А Н П Е Т Ч Ч

$N(s_i)$ 19 10 5 16 18 3 1 15 17 6 20 25

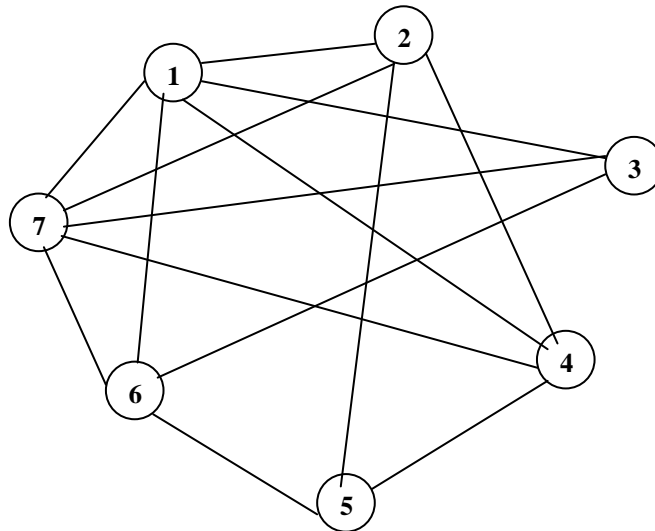
Y =

	19	10	5	16	18	3	1
19	0	9	14	3	1	16	18
10	9	0	5	6	8	7	9
5	14	5	0	11	13	2	4
16	3	6	11	0	2	13	15
18	1	8	13	2	0	15	17
3	16	7	2	13	15	0	2
1	18	9	4	15	17	2	0

A =

	1	2	3	4	5	6	7
1	0	1	1	1	0	1	1
2	1	0	0	1	1	0	1
3	1	0	0	0	0	1	1
4	1	1	0	0	1	0	1
5	0	1	0	1	0	1	0
6	1	0	1	0	1	0	1
7	1	1	1	1	0	1	0

G1:



Содержание

Лабораторная работа № 1. Подграфы и изоморфизм	4
Лабораторная работа № 2. Маршруты и связность в неориентированных графах	14
Лабораторная работа № 3. Поиск кратчайших маршрутов	21
Лабораторная работа № 4. Деревья и остовы неографов	29
Лабораторная работа № 5. Циклы и обходы	36
Лабораторная работа № 6. Ориентированные графы или орграфы	45
Лабораторная работа № 7. Плоские и планарные графы	59
Лабораторная работа № 8. Раскраска графов	71
Лабораторная работа № 9. Способы задания конечных автоматов	78
Лабораторная работа № 10. Минимизация конечных автоматов	91
Литература	97
Приложение А. Алгоритм генерации варианта	98