Томский межвузовский центр дистанционного образования

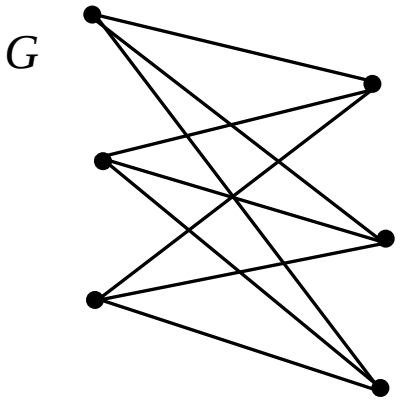
Н.В. Пермякова

СПЕЦГЛАВЫ МАТЕМАТИКИ

Часть 2

Теория графов

Учебное пособие



ТОМСК – 2002

Министерство образования Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Н.В. Пермякова

СПЕЦГЛАВЫ МАТЕМАТИКИ

Часть 2

Теория графов

Учебное пособие

2002

Пермякова Н.В.
Спецглавы математики. Часть 2. Теория графов: Учебное пособие. – Томск: Томский межвузовский центр дистанционного образования, 2002. – 122 с.

Учебное пособие предназначено для студентов, обучающихся по специальности 22020 «Автоматизация обработки информации и управления» с использованием дистанционных образовательных технологий. В данном учебном пособии рассмотрено пять разделов теории графов. Разделы содержат примеры решения задач, контрольные работы по дисциплине.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **5**

ГЛАВА 1. Основные понятия и определения **6**
1.1. Введение **6**
1.2. Операции над графами **9**
1.3. Специальные виды графов **11**
1.4. Неориентированные графы **13**
1.5. Ориентированные графы **14**
1.6. Бинарные отношения и ориентированные графы **15**
1.7. Решение задачи 4 контрольной работы №3 ..... **19**
1.8. Контрольные вопросы и упражнения **20**

ГЛАВА 2. Представление графов в ЭВМ **21**
2.1. Матричные представления графов **21**
2.2. Структура смежности и список ребер **23**
2.3. Решение задач 1, 2 контрольной работы №3 ..... **24**
2.4. Контрольные вопросы и упражнения **28**

ГЛАВА 3. Изоморфизм и планарность графов **29**
3.1. Изоморфизм **29**
3.2. Плоские и планарные графы **30**
3.3. Решение задач 3, 5 контрольной работы №3 ..... **31**
3.4. Контрольные вопросы и упражнения **33**

ГЛАВА 4. Маршруты на графах **35**
4.1. Основные понятия и определения **35**
4.1.1. Основные понятия и определения для неорграфа **35**
4.1.2. Основные понятия и определения для орграфа **37**
4.2. Понятие достижимости и связности **39**
4.2.1. Понятие достижимости и связности для неорграфа **39**
4.2.2. Понятия достижимости и связности для орграфов **40**
4.3. Матрицы достижимости, контрдостижимости, взаимодостижимости **43**
4.3.1. Булевы матрицы **43**
4.3.2. Матрицы достижимости, контрдостижимости и взаимодостижимости **44**
4.3.3. Алгоритмы вычисления матрицы достижимости **46**
4.3.4. Выделение компонент связности **49**
4.4. Метрики связного неорграфа **51**
4.5. Обход графа **53**

4.6. Минимальные маршруты на связных неорграфах **62**
4.7. Эйлеровы маршруты **73**
4.8. Решение задач 1-4 контрольной работы №4 ..... **75**
4.9. Контрольные вопросы и упражнения **79**

ГЛАВА 5. Деревья и циклы **81**
5.1. Основные определения **81**
5.2. Свойства деревьев **81**
5.3. Кодирование деревьев **83**
5.4. Обходы дерева **87**
5.5. Цикломатическое число **90**
5.6. Остовные деревья **93**
5.7. Базис независимых циклов **103**
5.8. Решение задачи 5 контрольной работы №4 ..... **106**
5.9. Контрольные вопросы и упражнения **108**

СПИСОК ЛИТЕРАТУРЫ **109**

ПРИЛОЖЕНИЕ 1. Графический материал к вариантам контрольных работ **110**
ПРИЛОЖЕНИЕ 2. Контрольная работа № 3 ..... **114**
ПРИЛОЖЕНИЕ 3. Варианты задачи 2 контрольной работы № 3 ..... **116**
ПРИЛОЖЕНИЕ 4. Контрольная работа № 4 ..... **118**
ПРИЛОЖЕНИЕ 5. Варианты задачи 4 контрольной работы № 4 ..... **120**

ВВЕДЕНИЕ

Дисциплина «Теория графов» является второй частью курса «Специальные главы математики» для студентов дистанционного обучения специальности 22020 «Автоматизация обработки информации и управления». В данном учебном пособии рассмотрены пять разделов теории графов, которые содержат примеры решения задач из контрольных работ по дисциплине. Варианты контрольных работ выбираются по общим правилам.

ГЛАВА 1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

1.1. Введение

Будем называть графом любую совокупность точек и линий, соединяющих эти точки. Причем не важно, какой линией соединены эти точки – прямой или криволинейной, длинной или короткой. Важно лишь соединены ли какие-то произвольно выбранные точки.

Будем называть точки графа ***вершинами***, а линии их соединяющие ***ребрами***.

В дальнейшем для обозначения вершин будем использовать строчные латинские буквы *x, y, z* или строчную латинскую букву *x* с индексом - x_i, для обозначения ребер - строчную латинскую букву *v* или букву *v* с индексом - v_i. Так же для обозначения ребра используют неупорядоченную пару (x_i,x_j) - ребро между вершинами x_i и x_j. В некоторых случаях для обозначения вершин будем использовать арабские цифры, для обозначения ребер – римские цифры.

Графом будем называть упорядоченную пару *G(X,V)*, где *X* – ***множество вершин,*** а *V* **– *множество ребер.*** Мощность множества *X* обозначим как *n(X)*, мощность множества *V* – *m(V)*.

В графе *G(X,V)* (рис. 1.1) пять вершин и шесть ребер, т.е. *n(X)=5*, а *m(V)=6*. Будем далее обозначать любой граф латинской прописной буквой *G* или буквой *G* с индексом.

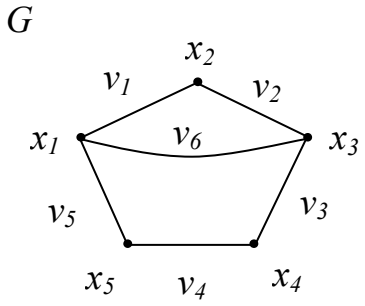


Рис. 1.1. **Граф *G(X,V)***

Граф можно задать перечислением множеств вершин и ребер.

Для графа *G* (рис 1.1) множества *X* и *V*:

$$X = \{x_1, x_2, x_3, x_4, x_5\}$$

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6,\} \text{ или}$$

$$V = \{(x_1,x_2),(x_2,x_3),(x_3,x_4),(x_4,x_5),(x_5,x_1),(x_1,x_3)\}.$$

Будем говорить, что для ребра v_1 графа *G* (рис. 1.1) вершины x_1 и x_2 являются ***концевыми*** вершинами.

Рассмотрим граф G (рис. 1.2). Ребра графа v_1 и v_2 имеют по одной концевой вершине. Такие ребра называются ***петлями***. Другими словами - ребро, начинающееся и заканчивающееся в одной и той же вершине, называется ***петлей***.

Ребра v_5 и v_6 графа G (рис. 1.2) имеют одинаковые концевые вершины, такие ребра будем называть ***кратными*** ребрами.

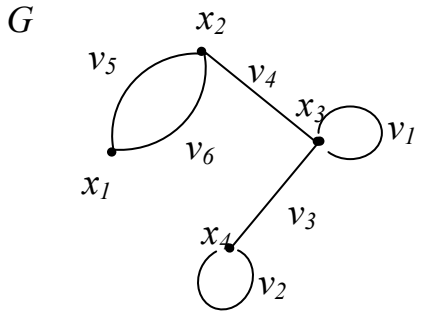


Рис. 1.2. **Псевдограф**

Граф с петлями и кратными ребрами называется ***псевдографом*** (рис. 1.2).
Граф без петель (рис.1.3) называется ***мультиграфом***.

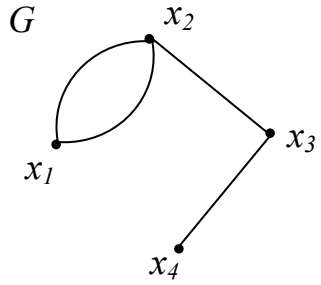


Рис. 1.3. **Мультиграф**

Граф без кратных ребер и петель (рис. 1.4) будем называть ***графом***.

Во всех ранее рассмотренных примерах (рис. 1.1 – 1.4) пары (x_i, x_j), $x_i, x_j \in X$ - неупорядоченные пары, иначе говорят, что ориентация ребра не задана.

Граф, для ребер которого не задана ориентация, называется ***неориентированным графом (неорграфом)***.

В некоторых графах задают направление движения по ребрам от вершины к вершине (задают ориентацию ребра) – такой граф называется ***ориентированным графом*** или ***орграфом*** (рис. 1.5).

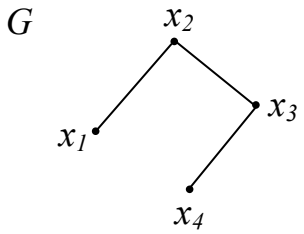


Рис. 1.4. **Граф**

В ориентированных графах ребра описываются упорядоченными парами. В графе G (рис. 1.5) ребро v_1 направлено от вершины x_1 к вершине x_2 - упорядоченная пара (x_1, x_2).

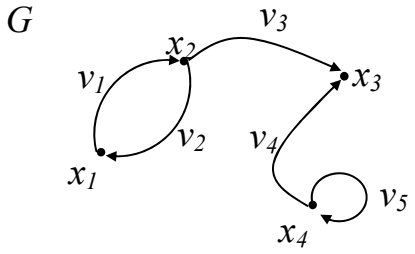


Рис. 1.5. **Ориентированный граф**

Запишем множества вершин и ребер орграфа G (рис. 1.5). Множество вершин:

$$X=\{x_1,x_2,x_3,x_4\}.$$

Множество ребер:

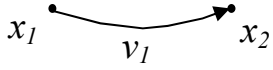
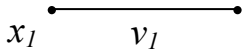
$$V=\{(x_1,x_2),(x_2,x_1),(x_2,x_3),(x_3,x_4),(x_4,x_4)\} \text{ или}$$

$$V=\{v_1,v_2,v_3,v_4,v_5\}.$$

В дальнейшем будем называть ребра орграфа ***дугами***. В таблице 1.1 представлены основные различия в обозначениях и определениях орграфов и неорграфов.

Таблица 1.1

Основные различия между орграфами и неорграфами

Орграф	Неорграф
 дуга (x_1,x_2) вершина x_1 - начало дуги v_1 вершина x_2 - конец дуги v_1	 ребро (x_1,x_2) или ребро (x_2,x_1) вершины x_1 и x_2 - концевые вершины ребра v_1

1.2. Операции над графами

Будем называть ***подграфом*** $G(X,V)$ граф $G'(X',V')$ такой, что множество X' является подмножеством множества X, а множество V' - подмножеством множестваV. Т.е. $X' \subseteq X$, $V' \subseteq V$.

Граф G' (рис. 1.6) подграф графа G (рис. 1.6).

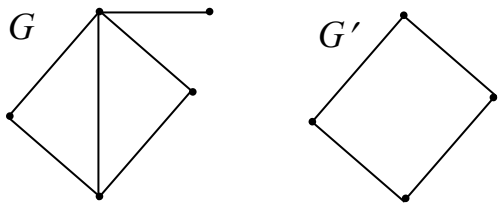


Рис. 1.6. **Граф *G* и его подграф *G′***

Собственным подграфом графа $G(X,V)$ будем называть подграф $G'(X',V')$ отличный от графа G (не совпадающий с графом G). В этом случае выполняется операция строгого включения $X' \subset X$ или $V' \subset V$.

Граф G' (рис. 1.6) является и собственным подграфом графа G (рис. 1.6).

Собственный подграф $G'(X',V')$ графа $G(X,V)$такой, что множества вершин графов G' и G совпадают, $X'=X$, а $V' \subset V$ называется ***суграфом*** (рис. 1.7).

Опишем процедуры удаления ребра и вершины из графа.

Операция ***удаления ребра*** подразумевает разрыв связи между какими-либо вершинами. Сами вершины в данном случае остаются неизменными.

Граф G' получен из графа G путем удаления ребер v_1 и v_2 (рис. 1.8).

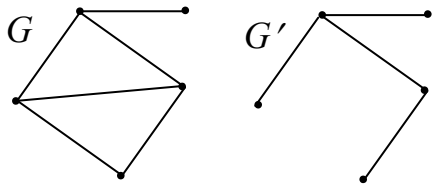


Рис. 1.7. **Граф G и его суграф G'**

Операция ***удаления вершины*** влечет за собой удаление всех ребер, для которых эта вершина является концевой.

Граф G'' получен из графа G путем удаления вершин x_1 и x_4 (рис. 1.8).

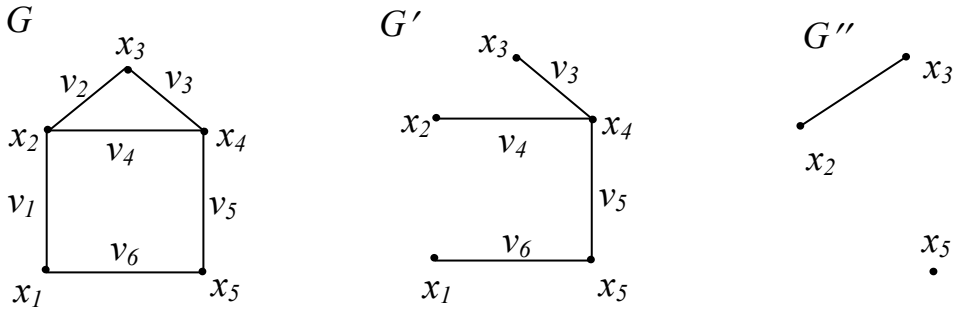


Рис. 1.8. **Операции удаления из графа ребра и вершины**

В графе G'' (рис. 1.8) вершина x_5 не связана ни с какой другой вершиной графа. Такие вершины называются ***изолированными***.

Вершина, являющаяся концевой для одного ребра, называется ***висячей***.

В графе G (рис. 1.8) висячих вершин нет. В графе G' (рис. 1.8) висячие вершины x_3, x_2 и x_1.

Еще одна операция над графами – стягивание графов. Операция стягивания это видоизменение графа, при котором уменьшается количество вершин, но ребра, концевую вершину которых удаляют, не убираются, а «склеиваются» (рис. 1.9).

Граф G стянут к графу G' (рис. 1.9). В этом примере удалена вершина x_2, «склеены» ребра v_1 и v_2.

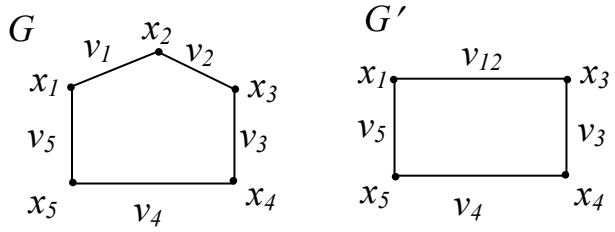


Рис. 1.9. **Стягивание графов**

1.3. Специальные виды графов

Некоторые виды графов имеют собственные названия.

Граф, состоящий только из изолированных вершин, называется ***пустым графом*** или ***нуль графом***. Множество вершин пустого графа $X \neq \varnothing$, т.е. в графе обязательно должна быть хотя бы одна вершина. Множество ребер пустого графа есть пустое множество, $V=\varnothing$ (рис. 1.10).

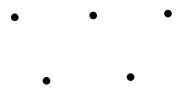


Рис. 1.10. **Пустой граф**

Пустой граф является собственным подграфом любого непустого графа. Он может получиться, например, путем удаления всех ребер из исходного графа.

Отдельное название получили и следующие типы графов.

Граф, любая вершина которых связана ребрами с остальными вершинами графа, называется ***полным*** графом.

Полные графы принято обозначать буквой K с индексом - K_n, где n - количество вершин графа (рис. 1.11).

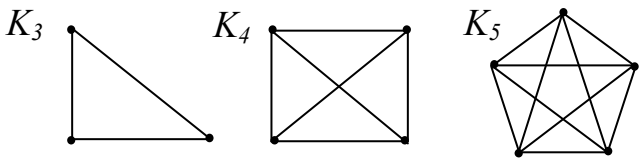


Рис. 1.11. **Полные графы**

Очевидно, что полный граф K_n - это граф с максимальным количеством ребер среди всех возможных графов на n вершинах. Максимальное количество ребер для графа на n вершинах:

$$M(G_n)=\frac{n\cdot(n-1)}{2}.$$

Рассмотрим граф $G(X,V)$. Пусть множество вершин X разделено на два непересекающихся подмножества X_1 и X_2, $X_1 \cup X_2 = X$ и $X_1 \cap X_2 = \varnothing$, т.е., вершины, принадлежащие одному множеству, не соединены между собой. Такие графы называются ***двудольными*** (рис. 1.12). Двудольные графы обозначают $G_{l\,k}$, где $l=|X_1|$, $k=|X_2|$.

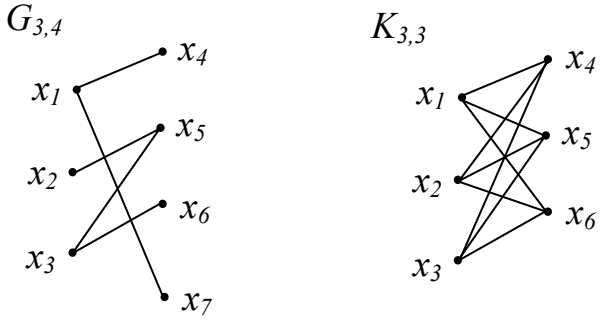


Рис. 1.12. **Двудольные графы**

Граф $G_{3,4}$ (рис. 1.12) - двудольный граф с множествами $X_1=\{x_1,x_2,x_3\}$ и $X_2=\{x_4,x_5,x_6,x_7\}$.

Двудольный граф, в котором все вершины множества X_1 связаны со всеми вершинами множества X_2, называют ***полным двудольным*** графом. Для обозначения таких графов используют букву K с двумя индексами – $K_{l,k}$. Граф $K_{3,3}$ (рис. 1.12) - полный двудольный граф с множествами вершин $X_1=\{x_1,x_2,x_3\}$ и $X_2=\{x_4,x_5,x_6\}$.

Чтобы перейти к следующему виду специальных графов, представим себе сеть улиц города в виде графа. Вершинами такого графа будут пересечения улиц, ребрами – сами улицы. Иногда с помощью таких графов решают задачи, в которых важен не только факт связи между двумя вершинами, но и расстояние между этими вершинами. В таких случаях, каждому ребру графа приписывают какое-то число w, которое может характеризовать, например, протяженность ребра или стоимость прохождения по ребру.

Число w называют ***весом*** ребра.

Графы, ребрам которых поставлены в соответствие веса *w*, называются ***нагруженными***, или ***взвешенными*** графами (рис. 1.13).

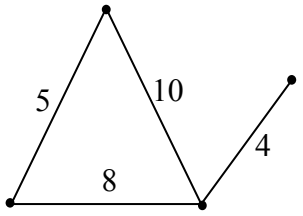


Рис. 1.13. **Нагруженный граф**

1.4. Неориентированные графы

Ранее уже отмечалось, что существуют некоторые различия в определениях для орграфов и неорграфов. Поэтому разделим следующие определения для орграфов и неорграфов в два подраздела.

Будем говорить, что две вершины неорграфа ***смежны***, если они являются концами одного ребра.

В графе *G* (рис. 1.14) смежные пары вершин: x_1 и x_2, x_2 и x_4, x_4 и x_3.

Два ребра неорграфа называются ***смежными***, если они имеют общую концевую вершину.

В графе *G* (рис. 1.14) смежные пары ребер: v_1 и v_2, v_2 и v_4 , v_4 и v_3 .

Вершина x ***инцидентна*** ребру v, если x - концевая вершина ребра v.

В графе *G* (рис. 1.14) вершина x_1 инцидентна ребрам v_1 и v_6, вершина x_2 инцидентна ребрам v_1 и v_2.

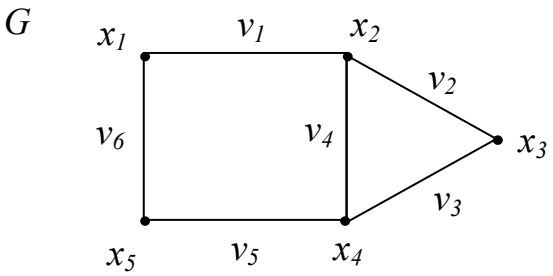


Рис. 1.14 **Неорграф *G***

Число ребер, инцидентных вершине *x*, называется ***степенью*** вершины $\rho(x)$.

В графе *G* (рис. 1.14) вершине x_4 инцидентны три ребра: v_5, v_4 и v_3, т.е. степень вершины x_4 равна трем, $\rho(x_4)=3$. Степень вершины x_5 - $\rho(x_5)=2$.

Пусть дан произвольный неорграф $G(X,V)$, $|X|=n$. Просуммировав степени всех вершин, получим число ребер, умноженное на два:

$$2m = \sum_{i=1}^{n} \rho(x_i) \qquad (1.1)$$

Удвоенное число ребер получилось из-за того, что каждое ребро учитывалось в сумме дважды, в степени каждой из двух концевых вершин. Из формулы (1.1) следует, что сумма степеней всех вершин – четное число.

Теорема 1.1. Число вершин нечетной степени четно.

Доказательство. Удалим из суммы (1.1) все четные степени. После этого число, стоящее в левой части суммы (1.1) останется четным. Следовательно, число вершин нечетной степени четно.

1.5. Ориентированные графы

В орграфе две вершины x и y называются ***смежными***, если вершина x - начало дуги v, а вершина y - конец дуги v.

В графе G (рис. 1.15) вершина x_1 смежна вершине x_2, а вершина x_4 не смежна вершине x_3.

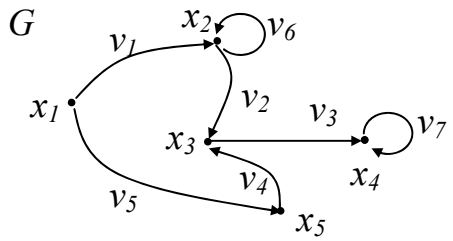


Рис. 1.15. **Ориентированный граф *G***

Вершина x ***инцидентна*** дуге v, если вершина x есть начало или конец дуги v.

В графе G (рис. 1.15) вершина x_1 инцидентна дугам v_1 и v_5, вершина x_2 инцидентна дугам v_1, v_2, v_6.

Число дуг, для которых вершина x, является началом, называется ***полустепенью исхода*** вершины x и обозначается $\rho^+(x)$.

Число дуг, для которых вершина x является концом, называется ***полустепенью захода*** вершины x и обозначается $\rho^-(x)$.

В графе G (рис. 1.15) вершина x_1 - начало дуг v_1 и v_5, полустепень исхода вершины x_1 : $\rho^+(x_1)=2$, вершина x_1 не является концом ни одного ребра, следовательно, полустепень захода вершины x_1 : $\rho^-(x_1)=0$.

Для вершины x_4: $\rho^+(x_4)=1$, $\rho^-(x_4)=2$.

1.6. Бинарные отношения и ориентированные графы

Пусть на множестве X задано бинарное отношение R, тогда $R=\{(x,y)|\ xRy,\ x,y \in X\}$.

В теории множеств уже отмечалось, что бинарные отношения могут представляться с помощью ориентированных графов. В этом случае множество X, на котором задано бинарное отношение, это множество вершин орграфа.

Бинарные отношения характеризуются своими свойствами.

Бинарное отношение называется ***рефлексивным***, если пара $(x,x)\in R$, $\forall x\in X$ или xRx, $\forall x\in X$ (рис. 1.16-*а*).

Ориентированный граф, представляющий рефлексивное отношение обязательно имеет петли при каждой вершине.

Бинарное отношение называется ***антирефлексивным***, если пара $(x,x)\notin R$, $\forall x\in X$.

Орграф, представляющий такое отношение не имеет ни одной петли (рис. 1.16-*б*).

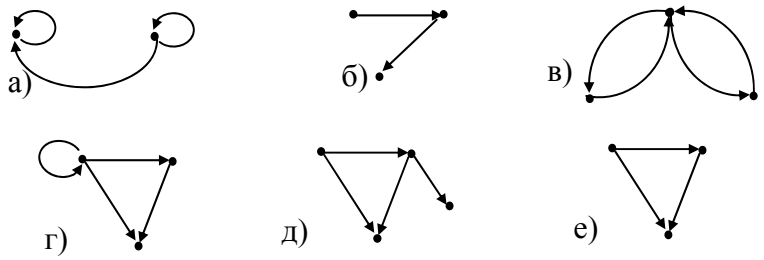


Рис. 1.16. **Орграфы и бинарные отношения**:
а) рефлексивный орграф; б) антирефлексивный орграф;
в) симметричный орграф; г) антисимметричный орграф;
д) несимметричный орграф; е) транзитивный орграф.

Бинарное отношение называется ***симметричным***, если выполняется условие $xRy \Leftrightarrow yRx$, $\forall x,y\in X$.

Если в орграфе G вершина x смежна вершине y, и вершина y смежна вершине x, причем это условие выполняется для всех пар смежных вершин, то G описывает симметричное отношение (рис. 1.16-*в*).

Бинарное отношение называется ***антисимметричным***, если выполняется условие $(x,y) \in R$ и $(y,x) \in R \Leftrightarrow x = y, \quad \forall\, x,y \in X$.

В орграфе G, представляющем такое отношение нет ни одной пары вершин x и y, такой, что, если x смежна вершине y, то и вершина y смежна вершине x, но допускаются петли (рис. 1.16 - *г*).

Бинарное отношение называется ***несимметричным***, если выполняется условие $(x,y) \in R \Leftrightarrow (y, x) \notin R, \quad \forall\, x,y \in X$.

В орграфе G, представляющем такое отношение нет ни одной пары вершин x и y, такой, что, если x смежна вершине y, то и вершина y смежна вершине x, и нет ни одной петли (рис. 1.16-*д*).

Бинарное отношение называется ***транзитивным***, если выполняется условие $(x,y) \in R$ и $(y,z) \in R \Leftrightarrow (x,z) \in R \quad \forall\, x,y,z \in X$.

По определению транзитивности, в орграфе, представляющем транзитивное бинарное отношение, для любых трех его вершин x, y и z должно выполняться условие: если вершина x есть начало дуги v, а вершина y - конец дуги v и если вершина y - начало дуги w, а вершина z - конец дуги w, то в орграфе есть дуга u, для которой вершина x - начало, а вершина z - конец (рис. 1.16-*е*). Дуга u называется ***транзитивно замыкающей*** дугой.

ПРИМЕР 1. Дано множество $X=\{-2,-1,1,2,3,4\}$. На множестве X задано бинарное отношение $M=\{(x,y) \mid |x| = |y|, x,y \in X\}$. Отношение M рефлексивно, симметрично, транзитивно. Действительно, по свойству рефлексивности, для всех элементов множества X $|x|=|x|$. По свойству симметричности, для всех элементов множества X, если $|x|=|y|$, то и $|y|=|x|$. По свойству транзитивности, если $|x|=|y|$ и $|y|=|z|$ то и $|x|=|z|$. Отношение M - отношение эквивалентности.

Зададим отношение M перечислением пар:

$M=\{(-2,2),(-2,2),(-1,-1),(-1,1),(1,1),(1,-1),(2,2),(2,-2),(3,3),(4,4)\}$.

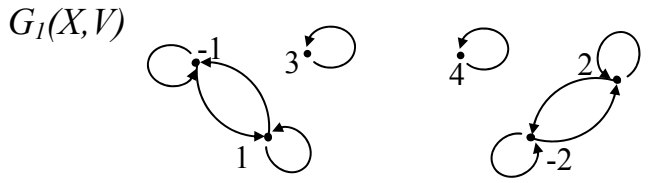


Рис. 1.17. **Орграф отношения *M***

Зададим отношение M орграфом $G_1(X,V)$ (рис. 1.17). В графе G_1 шесть вершин X={-2,-1,1,2,3,4} и десять ребер, т.е. столько же, сколько пар в отношении M. Будем строить дуги орграфа следующим образом: если пара $(x,y) \in M$, то ребро $(x,y) \in V$, т.е. множества ребер V и множество пар отношения M равны.

В графе $G_1(X,V)$ (рис.1.17) есть петли при всех вершинах, для всех вершин орграфа выполняется свойство симметричности и свойство транзитивности.

ПРИМЕР 2. Дано X={0,1,2,3}. На множестве X задано бинарное отношение $K=\{(x,y)\ ||x-y| \geq y,\ x,y \in X\}$. Отношение K не рефлексивное и не антирефлексивное, не симметричное, не несимметричное и не антисимметричное, не транзитивное. Перечислим элементы отношения:

K={(0,0), (0,1), (0,2), (0,3), (1,0), (2,0), (2,1), (3,0), (3,1)}.

Граф отношения K представлен на рис. 1.18.

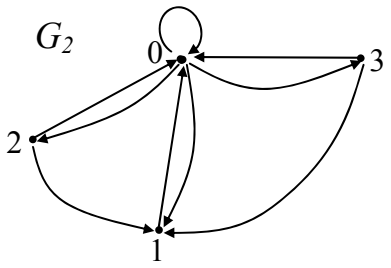


Рис. 1.18. **Орграф отношения *K***

В орграфе G_2 нет петель при вершинах 2,3 и 1, следовательно, орграф G_2 - не рефлексивный, но есть петля при вершине 0, следовательно, орграф G_2 и не антирефлексивный.

Вершина 2 смежна вершине 1, но вершина 1 не смежна вершине 2, следовательно, орграф не симметричный.

Орграф G_2 также не является антисимметричным, т.к. для вершин 0 и 2, 0 и 1, 0 и 3 выполняется условие симметричности, а $0 \neq 1$, $0 \neq 2$ и $0 \neq 3$. Орграф G_2 не обладает свойствами несимметричности, т.к. выполняется условие симметричности для некоторых вершин.

Орграф G_2 не является транзитивным, потому что для вершин 0, 2 и 3 не выполняется условие транзитивности: вершина 2 смежна вершине 0, вершина 0 смежна вершине 3, но вершина 2 не смежна вершине 3.

ПРИМЕР 3. Дано множество X={1,2,3,4}. На множестве X задано бинарное отношение P={(x,y) |$x \le y$, $x,y \in X$}. Отношение P рефлексивно, антисимметрично, транзитивно, т.е. отношение P - отношение частичного порядка.

Рассмотрим граф G_3 (рис. 1.19). Граф G_3 имеет петли при каждой вершине, условие симметричности для вершин x, y выполняется лишь в случае, если $x=y$. Для всех вершин графа выполняется условие транзитивности.

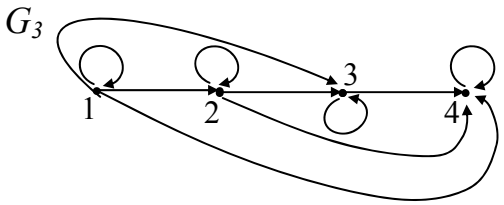


Рис. 1.19. **Орграф отношения *P***

Обратите внимание на особенности орграфов, представляющих бинарные отношения частичного порядка: наличие петель при каждой вершине, наличие транзитивно замыкающих дуг, одинаково направленная ориентация дуг.

Добавим в произвольном орграфе G все транзитивно замыкающие дуги. Полученный таким образом орграф называется ***транзитивным замыканием орграфа G*** (рис. 1.20).

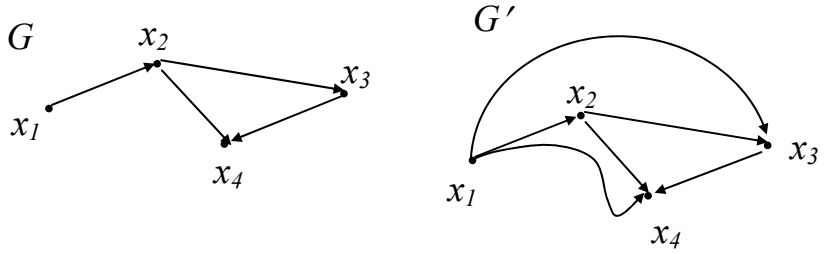


Рис. 1.20. **Граф G и его транзитивное замыкание G'**

Опишем построение транзитивного замыкания графа G.

Вершина x_1 смежна вершине x_2, вершина x_2 смежна вершине x_3, следовательно, необходимо добавить транзитивно замыкающую дугу (x_1,x_3). Вершина x_1 смежна вершине x_2, вершина x_2 смежна вершине x_4, следовательно, необходимо добавить транзитивно замыкающую дугу (x_1,x_4). Транзитивное замыкание построено.

Проверьте самостоятельно, что построены все транзитивно замыкающие дуги.

1.7. Решение задачи 4 контрольной работы №3

Задача 4. Записать бинарное отношение, заданное графом G (рис. 1.20). Определить, какими свойствами оно обладает.

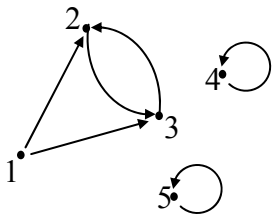


Рис. 1.20. **Граф *G***

Пронумеруем вершины графа произвольным образом. Запишем бинарное отношение представленное графом G перечислением, считая, что элементы множества X - вершины графа, тогда элементами бинарного отношения G_b будут дуги орграфа $G_b = \{(4,4), (5,5), (1,2), (1,3), (2,3), (3,2)\}$.

Отношение G_b не обладает свойством рефлексивности, т.к. не выполняется условие $(x,x) \in R, \forall x \in X$ или $xRx, \forall x \in X$, пары (1,1), (2,2), (3,3) $\notin G$. Не у всех вершин графа есть петли.

Отношение G_b не обладает свойством антирефлексивности, т.к. не выполняется условие $(x,x) \notin R, \forall x \in X$, пары (4,4) и (5,5) $\in G$. Свойство антирефлексивности предполагает отсутствие петель в графе.

Отношение не обладает свойством симметричности, т.к. не выполняется условие $xRy \Leftrightarrow yRx, \forall x,y \in X$. В графе G есть пары (1,3) и (1,2), но нет пар (3,1) и (2,1).

Отношение G_b не обладает свойством антисимметричности, т.к. не выполняется условие $(x,y) \in R$ и $(y,x) \in R \Leftrightarrow x=y, \forall x,y \in X$. В графе есть дуга (1,2) и дуга (2,1), но $2 \neq 1$.

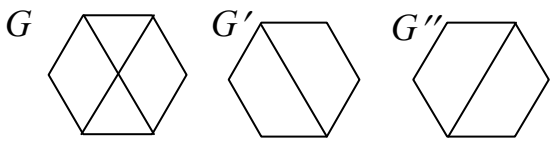
Отношение G_b не обладает свойством несимметричности, т.к. не выполняется условие $(x,y) \in R \Leftrightarrow (y,x) \notin R, \forall x,y \in X$, потому что в графе есть и дуга (1,2) и дуга (2,1).

Отношение G_b не обладает свойством транзитивности, т.к. не выполняется условие $(x,y) \in R$ и $(y,z) \in R \Leftrightarrow (x,z) \in R, \forall x,y,z \in X$. Например, в графе есть дуга (3,2) и дуга (2,3), но нет транзитивно замыкающей дуги (3,3).

1.8. Контрольные вопросы и упражнения

1. Сформулируйте различия между псевдографом, мультиграфом и графом.

2. Какой граф является собственным подграфом графа G?



3. Будет ли пустой граф, образованный из графа G, удалением всех вершин суграфом G?

4. Сколько ребер содержит граф K_6?

5. Нарисуйте граф $K_{3,2}$.

6. Запишите все пары смежных вершин для графа на рис. 1.14.

7. Посчитайте полустепени захода и исхода для всех вершин графа на рис. 1.15.

8. Удалите из графа на рис. 1.18 вершину 1. Нарисуйте полученный граф.

9. Удалите из графа на рис. 1.17 ребро (2,3). Нарисуйте полученный граф.

10. Нарисуйте произвольный граф на четырех вершинах, который бы представлял бинарное отношение, обладающее свойствами рефлексивности, антисимметричности и не обладающее свойством транзитивности.

ГЛАВА 2. ПРЕДСТАВЛЕНИЕ ГРАФОВ В ЭВМ

2.1. Матричные представления графов

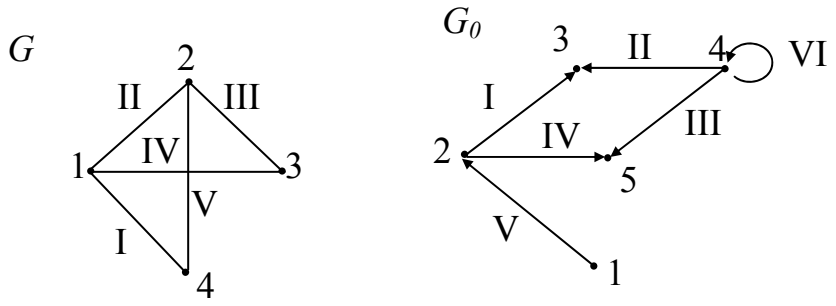
В этом разделе будем обозначать вершины графов арабскими цифрами 1,2,…,n , а ребра и дуги римскими I, II, III, …

Будем называть ***матрицей смежности*** графа, матрицу $A(n\times n)$, элементы которой $a_{i,j}$ формируются следующим образом:

$$a_{ij}=\begin{cases}1, & \text{если вершина } i \text{ смежна вершине } j;\\ 0, & \text{в противном случае.}\end{cases}$$

ПРИМЕР 1. Построить матрицы смежности для графов G и G_0 (рис. 2.1).

Построим матрицы A и A_0 соответственно:



$$A=\begin{bmatrix}0&1&1&1\\1&0&1&1\\1&1&0&0\\1&1&0&0\end{bmatrix}\qquad A_o=\begin{bmatrix}0&1&0&0&0\\0&0&1&0&1\\0&0&0&0&0\\0&0&1&1&1\\0&0&0&0&0\end{bmatrix}$$

Рис. 2.1. **Графы и их матрицы смежности**

Отметим, что для неорграфа матрица смежности всегда будет симметричной и элементы главной диагонали равны нулю.

Отметим так же, что число единиц в i-той строке (или столбце) матрицы смежности равно степени вершины с номером i.

Сумма единичных элементов матрицы смежности равна удвоенному числу ребер неорграфа.

Для матрицы смежности орграфа можно отметить следующее.

Если элемент главной диагонали матрицы смежности $a_{i,i}=1$, значит у вершины с номером i есть петля.

Сумма единичных элементов матрицы смежности равна числу ребер орграфа.

Сумма единичных элементов i-той строки матрицы смежности равна полустепени исхода i-той вершины орграфа $\rho^{+}(i)$.

Чему, в таком случае, будет равна сумма единичных элементов i-того столбца матрицы смежности?

Следующий матричный способ представления графов – ***матрица инцидентности*** $B(n\times m)$. Элементы матрицы формируются следующим образом:

а) для неорграфа -

$$b_{ij}=\begin{cases}1, \text{если вершина } i \text{ инцидентна вершине } j;\\ 0, \text{в противном случае.}\end{cases}$$

б) для орграфа -

$$b_{ij}=\begin{cases}1, & \text{если вершина } i \text{ - начало дуги } j;\\ -1, & \text{если вершина } i \text{ - конец дуги } j;\\ 2, & \text{если дуга } j \text{ - петля при вершине } i;\\ 0, & \text{если вершина } i \text{ не инцидентна дуге } j.\end{cases}$$

ПРИМЕР 2. Составим матрицы инцидентности B и B_0 для графов G и G_0 (рис. 2.1) соответственно.

$$B=\begin{bmatrix}1&1&0&1&0\\0&1&1&0&1\\0&0&1&1&0\\1&0&0&0&1\end{bmatrix};\qquad B_o=\begin{bmatrix}0&0&0&0&1&0\\1&0&0&1&-1&0\\-1&-1&0&0&0&0\\0&1&1&0&0&2\\0&0&-1&-1&0&0\end{bmatrix}.$$

Для матрицы инцидентности неорграфа можно отметить, что в любом столбце матрицы ровно два единичных элемента.

Если в j-том столбце матрицы инцидентности $a_{j,k}=1$ и $a_{j,l}=1$, то концевыми вершинами ребра j являются вершины k и l.

Сумма единичных элементов в строке i равна степени вершины i.

Для матрицы инцидентности орграфа отметим, что в любом столбце матрицы инцидентности либо ровно один единичный элемент и ровно один элемент равен –1, либо ровно один элемент равен двум, все остальные элементы столбца нулевые.

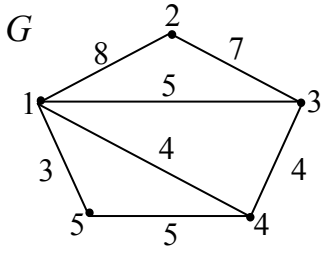
Если в j-том столбце матрицы инцидентности $a_{j,k}$=-1 и $a_{j,l}$=1, то вершина l - начало ребра j, а вершина k - конец ребра j.

Число единичных элементов i-той строки матрицы инцидентности равно $\rho^+(i)$, число элементов, равных –1 равно $\rho^-(i)$.

Для представления нагруженных графов используется ***матрица весов*** $W(n\times n)$. Элементы матрицы формируются следующим образом:

$$w_{ij}=\begin{cases}\text{вес ребра или дуги } (i,j), & \text{если } (i,j)\in V;\\ \infty, & \text{в противном случае.}\end{cases}$$

ПРИМЕР 3. Для графа G (рис. 2.2) составим матрицу весов W.



$$W=\begin{bmatrix}\infty & 8 & 5 & 4 & 3\\ 8 & \infty & 7 & \infty & \infty\\ 5 & 7 & \infty & 4 & \infty\\ \infty & \infty & 4 & \infty & 5\\ 3 & \infty & \infty & 5 & \infty\end{bmatrix}$$

Рис. 2.2. **Нагруженный граф и его матрица весов**

2.2. Структура смежности и список ребер

Структура смежности – каждой вершине графа в соответствие ставится список смежных вершин.

Структуры смежности программно реализуются массивом из n линейно связанных списков.

Список ребер орграфа представляет собой два одномерных массива, назовем их NR и KR, размерности m, элементы которых формируются следующим образом:

$nr_i \in V,\ i=\overline{1,m},\quad nr_i$ – начало i-ой дуги;

$kr_i \in V,\ i=\overline{1,m},\quad kr_i$ – конец i-ой дуги.

Список ребер неорграфа формируется по такому же правилу, но так как любая из двух вершин, инцидентных одному ребру может быть и началом и концом этого ребра, то каждое ребро записывается в список ребер дважды. Таким образом, размерность массивов *NR* и *KR* для неорграфа – *2m*.

ПРИМЕР 4. Построить структуру смежности и список ребер для графов *G* и G_0 (рис. 2.1).

Структура смежности графа *G*		Структура смежности графа G_0	
V	Список смежных вершин	*V*	Список смежных вершин
1:	2, 3, 4	1:	2
2:	1, 3, 4	2:	3, 5
3:	1, 2	3:	
4:	1, 2	4:	5, 3, 4
		5:	

Список ребер графа *G*: NR=(1,4,1,2,2,3,3,1,2,4).
KR=(4,1,2,1,3,2,1,3,4,2).

Список ребер графа G_0: NR_0=(2,4,4,2,1,4).
KR_0=(3,3,5,5,2,4).

От одной формы представления графов легко можно перейти к любой другой.

2.3. Решение задач 1-2 контрольной работы №3

Задача 1. Представить графы *G* и G_0 четырьмя способами.

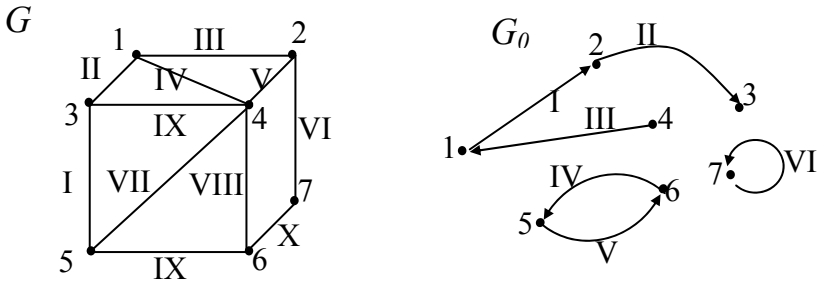


Рис. 2.3. **Неорграф *G* и орграф G_0**

Пронумеруем произвольно вершины и ребра графа G (рис. 2.3).

В графе G 7 вершин и 11 ребер, следовательно, матрица смежности графа G - матрица $A(7\times7)$, матрица инцидентности - $B(7\times11)$:

$$A(G)=\begin{bmatrix}0&1&1&1&0&0&0\\1&0&0&1&0&0&1\\1&0&0&1&1&0&0\\1&1&1&0&1&1&0\\0&0&1&1&0&1&0\\0&0&0&1&1&0&1\\0&1&0&0&0&1&0\end{bmatrix}\qquad B(G)=\begin{bmatrix}0&1&1&1&0&0&0&0&0&0&0\\0&0&1&0&1&1&0&0&0&0&0\\1&1&0&0&0&0&0&0&0&0&1\\0&0&0&1&1&0&1&1&0&0&1\\1&0&0&0&0&0&1&0&1&0&0\\0&0&0&0&0&0&0&1&1&1&0\\0&0&0&0&0&1&0&0&0&1&0\end{bmatrix}$$

Построим структуру смежности для неорграфа G. Это семь связных списков, вершине 1 смежны вершины 2, 3, 4 - 1: 2, 3, 4;

вершине 2 смежны вершины 1,4,7 - 2: 1,4,7;
вершине 3 смежны вершины 1, 4, 5 - 3: 1, 4, 5;
вершине 4 смежны вершины 1, 2, 3, 5,6 - 4: 1, 2, 3, 5,6;
вершине 5 смежны вершины 3, 4, 6 - 5: 3, 4, 6;
вершине 6 смежны вершины 4, 5, 7 - 6: 4, 5, 7;
вершине 7 смежны вершины 2, 6- 7: 2, 6.

Построим список ребер для неорграфа G. Так как в неорграфах одна вершина может быть и началом и концом ребра, то размерность массивов начальных и конечных вершин в списке ребер равна $2\cdot m=2\cdot11=22$.

$$NR=\{5,3,1,1,2,2,4,4,5,6,3,3,1,2,4,4,7,5,6,6,7,4\};$$
$$KR=\{3,1,2,4,4,7,5,6,6,7,4,5,3,1,1,2,2,4,4,5,6,3\}.$$

В графе G_0 7 вершин и 6 ребер, следовательно, матрица смежности $A(G_0)$, размерности (7×7), и матрица инцидентности $B(G_0)$, размерности (7×6):

$$A(G_0)=\begin{bmatrix}0&1&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&0&0&0\\1&0&0&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\0&0&0&0&0&0&1\end{bmatrix}\qquad B(G_0)=\begin{bmatrix}1&0&-1&0&0&0\\-1&1&0&0&0&0\\0&-1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&-1&1&0\\0&0&0&1&-1&0\\0&0&0&0&0&2\end{bmatrix}$$

Составим структуру смежности графа G_0:

Вершине 1 смежна вершина 2- 1: 2;

Вершине 2 смежна вершина 3- 2: 3;

У вершины 3 нет смежных вершин – 3: ;

Вершине 4 смежна вершина 1- 4: 1;

Вершине 5 смежна вершина 6- 5: 6;

Вершине 6 смежна вершина 5- 6: 5;

Вершине 7 смежна вершина 7- 7: 7;

Составим список ребер орграфа G_0. Размерность массивов начальных и конечных вершин в списке ребер равна m=6.

$$NR=\{1,2,4,5,6,7\};$$
$$KR=\{2,3,1,6,5,7\}.$$

Задача 2. Описать алгоритм и составить программу составления матрицы смежности по данному списку ребер.

Запишем алгоритм составления матрицы смежности по списку ребер. По определению, любой единичный элемент матрицы смежности $a_{i,j}$ определяет ребро между вершинами i и j. По определению списка ребер элемент NR_k есть начало k-того ребра, элемент KR_k есть конец k-того ребра. Будем просматривать элементы массивов NR, KR.

Пусть $NR_k = z$, $KR_k = l$, тогда элемент матрицы смежности $a_{z,l} = 1$. Программа решения задачи на языке PASCAL:

```
Program Matrix;
Uses Crt;
Type Mn_Ver = set of 1..25;                {* определение типа множества вершин *}
Var n : Byte;                               {* Количество вершин графа *}
   NR,KR : Array [1..100] of byte;          {* Список ребер *}
   A : Array[1..50,1..50] of byte;          {* Матрица смежности *}
   Ver : Mn_Ver;                            {* Множество вершин *}
   i,j,z,l : Byte;

Begin
 {* Ввод элементов списка ребер *}
{************************************************************}
 ClrScr;
 WriteLN('Введите количество элементов списка ребер (не более 50): ');
 ReadLN(z);
```

```
 For i := 1 To z Do
   Begin
   Write('Введите начало ребра: ');
   ReadLN(NR[i]);
   Write('Введите конец ребра: ');
   ReadLN(KR[i]);
   End;
{*****************************************************************}
 {* Печать списка ребер *}
{*****************************************************************}
 WriteLN('Список ребер:');
 Write('NR = (');
 For i := 1 To z Do
   Write(NR[i]:3);
 WriteLN(')');
 Write('KR = (');
 For i := 1 To z Do
   Write(KR[i]:3);
 WriteLN(')');
 {*****************************************************************}
Ver := [];                                          {* Множество вершин - пустое *}
 For i := 1 to z Do                                 {* Формирование *}
    Begin                                           {* множества    *}
    If not(NR[i] in Ver) Then Include(Ver,NR[i]);   {* вершин       *}
    If not(KR[i] in Ver) Then Include(Ver,KR[i]);
    End;
 n := 0;                                            {* Подсчет     *}
 For i := 1 To 25 Do                                {* количества  *}
   If i in Ver Then Inc(n)                          {* вершин      *}
              Else Break;
 Write('В графе ',n,' вершин(ы).');
 Writeln(' Нажмите любую клавишу для продолжения...');
 ReadKey;
 {* Обнуление матрицы смежности *}
 {*****************************************************************}
For i := 1 To n Do
     For j := 1 To n Do
         A[i,j] := 0;
 {*****************************************************************}
{* Формирование матрицы смежности *}
 For i := 1 To Z Do
   Begin
   z := NR[i];
   l := KR[i];
   A[z,l] := 1;
   End;
 {*****************************************************************}
{* Печать матрицы смежности *}
 WriteLN('Матрица смежности: ');
 For i := 1 To n Do
```

```
    Begin
    For j := 1 To n Do
       Write( A[i,j] : 3);
    WriteLN;
    End;
 {************************************************************}
Writeln(' Нажмите любую клавишу для окончания работы...');
 ReadKey;
End.
```

2.4. Контрольные вопросы и упражнения

1. Запишите алгоритм подсчета ребер в неорграфе, если граф задан матрицей смежности.

2. Запишите матрицы смежности и инцидентности для графов G и G_0 (рис. 2.5), произвольно занумеровав вершины и ребра графов.

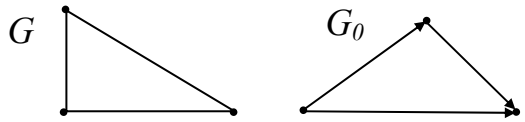


Рис. 2.5. **Неорграф G и орграф G_0**

3. Опишите алгоритм нахождения степеней всех вершин неорграфа, заданного структурой смежности.

4. Укажите правильные утверждения:

а) количество единичных элементов в i-том столбце матрицы инцидентности равно степени i-той вершины;

б) матрица смежности любого неорграфа есть симметричная матрица относительно главной диагонали;

в) матрица инцидентности любого орграфа – квадратная матрица;

г) в любом столбце матрицы инцидентности ровно два отличных от нуля элемента.

5. Какова будет матрица смежности орграфа, обладающего свойствами рефлексивности, симметричности, транзитивности.

6. Опишите алгоритм расчета числа ребер неорграфа, заданного структурой смежности.

7. Запишите структуру смежности и список ребер для графов G и G_0.

8. Как называется вершина, которой в матрице инцидентности соответствует строка, состоящая из нулевых элементов?

ГЛАВА 3. ИЗОМОРФИЗМ И ПЛАНАРНОСТЬ ГРАФОВ

3.1. Изоморфизм

Рассмотрим два графа G и G' (рис. 3.1). Графы имеют одинаковую структуру, это видно, если «растянуть» ребра графа G' в прямую линию. Единственным различием между двумя графами на рис. 3.1 осталась неодинаковая нумерация вершин.

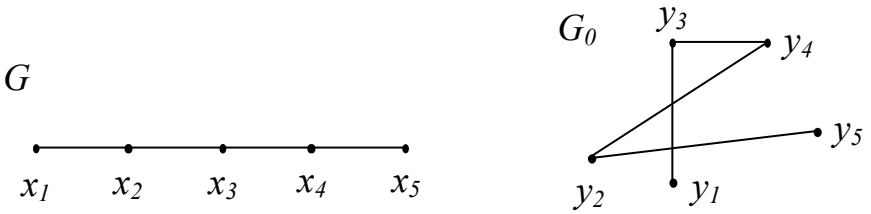


Рис. 3.1. **Изоморфные графы**

Построим соответствие $f : X \to Y$, где X - множество вершин графа G, множество Y - множество вершин графа G':

$$f(x_1)=y_1;\ f(x_2)=y_3;\ f(x_3)=y_4;\ f(x_4)=y_2;\ f(x_5)=y_5.$$

Соответствие $f : X \to Y$ - биекция; однозначное, функциональное отображение.

Графы $G(X,V)$ и $G'(Y,Z)$ называются ***изоморфными***, если существует биекция $f : X \to Y$, сохраняющая смежность. Поясним понятие «сохраняющая смежность»: если вершина x смежна вершине z в графе G, то и $f(x)$ смежна вершине $f(z)$ в графе G'.

Проверим для графов G и G' (рис. 3.1) сохраняет ли построенная биекция смежность.

Вершина x_1 смежна вершине x_2 и $f(x_1)=y_1$ смежна $f(x_2)=y_3$.
Вершина x_2 смежна вершине x_3 и $f(x_2)=y_3$ смежна $f(x_3)=y_4$.
Вершина x_3 смежна вершине x_4 и $f(x_3)=y_4$ смежна $f(x_4)=y_2$.
Вершина x_4 смежна вершине x_5 и $f(x_4)=y_2$ смежна $f(x_5)=y_5$.

Построенная биекция $f : X \to Y$ сохраняет смежность, следовательно, графы G и G' изоморфны.

Таким образом, изоморфизм – это не равенство самих графов, а равенство их структуры, формы.

Матрицы смежности и инцидентности изоморфных графов получаются одна из другой перестановкой строк и столбцов.

Рассмотрим это утверждение на примере графов G и G' (рис. 3.1)
Построим матрицы смежности графов G и G' :

$$A(G)=\begin{bmatrix}0&1&0&0&0\\1&0&1&0&0\\0&1&0&1&0\\0&0&1&0&1\\0&0&0&1&0\end{bmatrix} \qquad A(G')=\begin{bmatrix}0&0&1&0&0\\0&0&0&1&1\\1&0&0&1&0\\0&1&1&0&0\\0&1&0&0&0\end{bmatrix}$$

Переставим столбцы и строки в матрице $A(G')$ в следующем порядке:
1, 3, 4, 2, 5.

Переставим столбцы:

$$\begin{bmatrix}0&1&0&0&0\\0&0&1&0&1\\1&0&1&0&0\\0&1&0&1&0\\0&0&0&1&0\end{bmatrix}$$

Переставим строки:

$$\begin{bmatrix}0&1&0&0&0\\1&0&1&0&0\\0&1&0&1&0\\0&0&1&0&1\\0&0&0&1&0\end{bmatrix}$$

Полученная матрица равна матрице $A(G')$.

3.2. Плоские и планарные графы

Граф называется ***плоским***, если его ребра пересекаются лишь в вершинах (рис. 3.2).

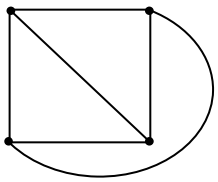


Рис. 3.2. **Плоский граф**

Граф G (рис.3.3) – плоский граф.
Граф G_1 (рис.3.3) – не плоский граф. Но граф G_1 изоморфен графу G, а значит, его можно представить плоским графом.

Графы, изоморфные плоским графам называются ***планарными***.

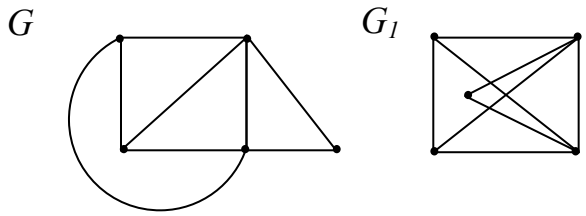


Рис. 3.3. **Плоский и планарный графы**

Не всякий граф можно представить в форме плоского графа. Следующий критерий планарности графа носит имя Понтрягина-Куратовского.

Если в графе можно выделить подграф K_5 или $K_{3,3}$, то такой граф не является планарным.

3.3. Решение задач 3, 5 контрольной работы №3

Задача 3. Проверить, изоморфны ли графы G_1 и G_2 графу G (рис. 3.4).

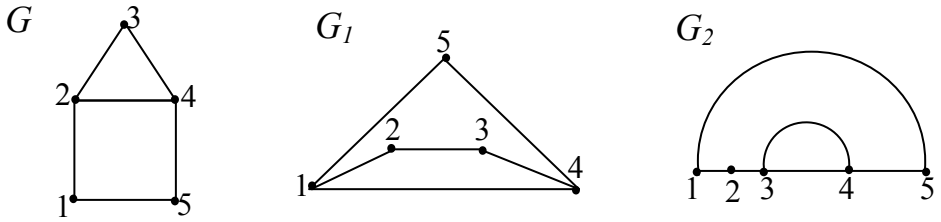


Рис. 3.4. **Изоморфизм графов**

Для того, чтобы два графа $G(X,V)$ и $G_1(X_1,V_1)$ были изоморфны, необходимо чтобы выполнялись равенства $|X|=|X_1|$, $|V|=|V_1|$. Проверим выполнение этих условий для графов G_1 и G_2.

$|X|=5$ $\quad |X_1|=5$ $\quad |X_2|=5$

$|V|=6$ $\quad |V_1|=6$ $\quad |V_2|=6$.

Далее сравним степени вершин в графах G_1 и G_2 со степенями вершин в графе G.

Граф G: $\rho(1)=2,\ \rho(2)=3,\ \rho(3)=2,\ \rho(4)=3,\ \rho(5)=2$.

Граф G_1: $\rho_1(1)=3,\ \rho_1(2)=2,\ \rho_1(3)=2,\ \rho_1(4)=3,\ \rho_1(5)=2$.

Граф G_2: $\rho_2(1)=2,\ \rho_2(2)=2,\ \rho_2(3)=3,\ \rho_2(4)=3,\ \rho_2(5)=2$.

У всех графов одинаковое количество вершин со степенью 2 и со степенью 3.

Построим соответствие $f: X \to X_1$ следующим образом:

$$f(1)=2,\ f(2)=1,\ f(3)=5,\ f(4)=4,\ f(5)=3.$$

Установленное соответствие является биекцией, сохраняющей смежность. Это легко увидеть, перерисовав граф G_1, как показано на рис. 3.5. Следовательно, графы G и G_1 изоморфны.

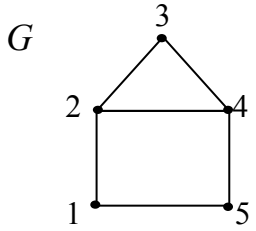


Рис. 3.5. **Граф G_1**

Рассмотрим теперь структуру графа G_2. Граф G_2 - мультиграф, в нем есть кратные ребра (между вершинами 3 и 4). Следовательно, граф G_2 имеет структуру, отличную от структуры графа G. Между множествами вершин этих графов невозможно построить биекцию, сохраняющую смежность. Вывод: графы G и G_2 неизоморфны.

Задача 5. Проверить, планарный ли граф G_1 (рис. 3.6).

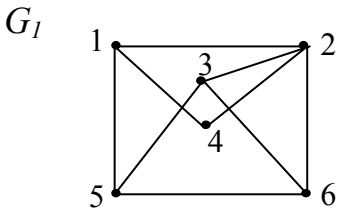


Рис. 3.6. **Граф G_1**

По критерию Понтрягина-Куратовского граф не планарный, если в нем можно выделить подграф K_5 или $K_{3,3}$. Проверим, можно ли выделить в графе G_1 подграф K_5.

Рассмотрим все возможные подграфы графа G_1, множество вершин которых состоит из пяти элементов.

Для этого из графа G_1 поочередно удаляется по одной вершине, всего можно рассмотреть шесть таких подграфов (рис. 3.7. *а-е*).

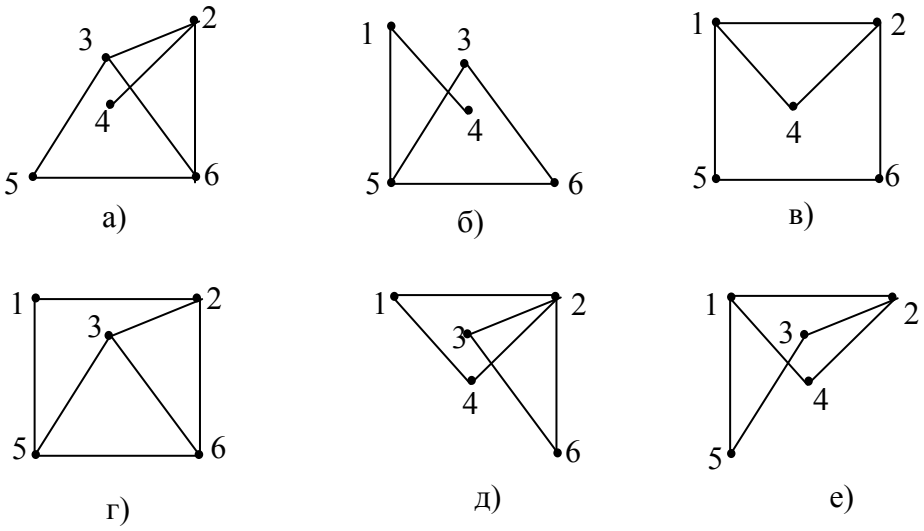


Рис. 3.7. **Подграфы графа G_1**:
а) удаление вершины 1; б) удаление вершины 2; в) удаление вершины 3; г) удаление вершины 4; д) удаление вершины 5; е) удаление вершины 6.

Ни один из построенных подграфов не является K_5. Таким образом, граф G_1 - планарный граф. Изоморфный ему плоский граф представлен на рис. 3.8.

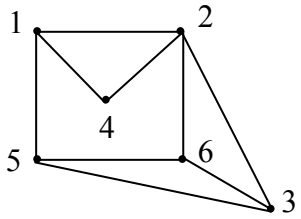


Рис. 3.8. **Плоский граф, изоморфный G_1**

В общем случае при проверке планарности графа с n вершинами необходимо проверить $C_n^5 = \frac{n!}{5!(n-5)!}$ подграфов с пятью вершинами. Для небольших графов возможно опустить выделение подграфов, изобразив плоский граф, изоморфный исходному.

3.4. Контрольные вопросы и упражнения

1. Всякий ли граф с n вершинами и n+2 ребрами будет планарным?

2. Выберите графы, изоморфные графу G (рис. 3.9).

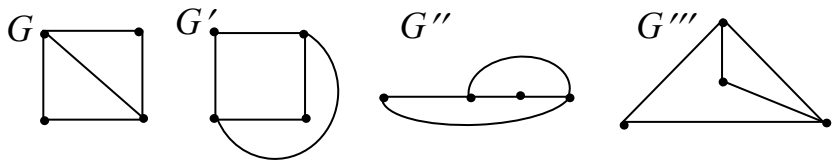


Рис. 3.9. **Упражнение 2**

3. Получится ли планарный граф, если из K_6 удалить одну вершину?

4. Выберите правильные по вашему мнению утверждения:

а) любой граф на четырех вершинах является планарным графом;
б) все графы с n вершинами и m ребрами изоморфны;
в) планарный граф всегда можно представить плоским графом;
г) любой граф с шестью вершинами и десятью ребрами является планарным графом.

5. Выберите среди графов на рис. 3.10 плоские и планарные графы.

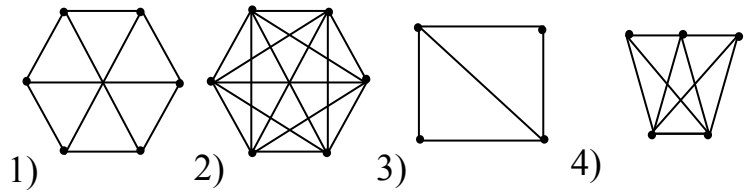


Рис. 3.10. **Упражнение 5**

6. Сколько подграфов необходимо построить, чтобы ответить на вопрос о планарности графа с $|X|=9$?

ГЛАВА 4. МАРШРУТЫ НА ГРАФАХ

4.1. Основные понятия и определения

4.1.1. Основные понятия и определения для неорграфа

Теория графов применяется в различных прикладных областях. Графом можно представить железнодорожные линии между городами, систему автодорог, распределенную компьютерную сеть, электрические схемы и т.д. Во многих прикладных областях решаются задачи нахождения маршрута между двумя точками или, говоря терминами теории графов, между двумя вершинами.

Маршрутом между двумя вершинами x и y неорграфа называется упорядоченная последовательность ребер и вершин, которые нужно пройти, чтобы попасть из вершины x в вершину y.

Запишем примеры маршрутов из вершины 1 в вершину 3 в графе G (рис.4.1- *а*)

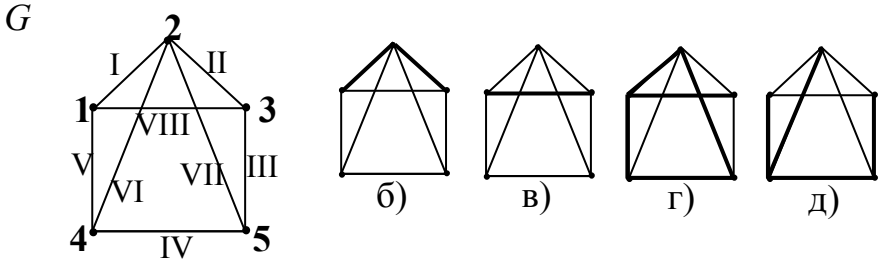


Рис. 4.1. **Маршруты на графах:**
а) неорграф *G*; б) маршрут *M1*; в) маршрут *M2*;
г) маршрут *M3*; д) маршрут *M4*

M1= 1, I, 2, II, 3.
M2= 1, VIII, 3.
M3= 1, V, 4, IV, 5, VII, 2, I, 1, VIII, 3.
M4= 1, V, 4, VI, 2, VI, 4, IV, 5, III, 3.

На рис. 4.1 *б – д*, записанные маршруты выделены более жирными линиями.

В записи маршрута для графа без кратных ребер и петель можно опустить ребра. Тогда маршруты *M1*, *M2*, *M3*, *M4* запишутся следующим образом:

M1= 1, 2, 3.

M2= 1, 3.
M3= 1, 4, 5, 2, 1, 3.
M4= 1, 4, 2, 4, 5, 3.

Так же можно опустить в записи маршрутов все вершины, оставив одни ребра. Запишите маршруты *M1*, *M2*, *M3*, *M4* самостоятельно.

Цепью будем называть маршрут, начальная и конечная вершины которого различны.

Маршруты *M1*, *M2*, *M3*, *M4*- цепи, т.к. начальная вершина маршрутов вершина 1, конечная вершина маршрута вершина 3.

Цепь, в которой нет повторяющихся вершин, т.е. каждая вершина пройдена один раз, называется ***простой*** цепью.

Маршруты *M1*, *M2*- простые цепи.

Цепь, в которой первая и последняя вершины совпадают, называется ***циклом***.

В графе *G* можно выделить следующие циклы:
C1= 1, I, 2, VI, 4, V, 1.
C2= 1, I, 2, VI, 4, IV, 5, IIV, 2, I, 1.
C3= 1, VIII, 3, III, 5, IV, 4, V, 1.
C4 = 2, VI, 4, IV, 5, VII, 2, II, 3, VIII, 1, I, 2.

На рис. 4.2 *а* - *г* записанные циклы выделены более жирными линиями.

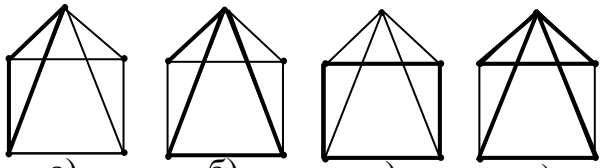


Рис. 4.2. **Циклы в графах:**
а) цикл *C1*; б) цикл *C2*; в) цикл *C3*; г) цикл *C4*

Цикл, образованный простой цепью называется ***простым*** циклом.

C1 и *C3* - простые циклы.

Длиной маршрута будем считать количество ребер, входящих в маршрут.

Длина цепи *M1* равна двум, длина цикла *C2* равна 5.

Если маршрут строится на нагруженном неорграфе, то длиной маршрута считается сумма весов ребер, входящих в маршрут.

Построим два маршрута на графе *G* (рис. 4.3).

$M1 = x_1x_2x_4$; $M2 = x_1x_3x_4$. Длина маршрута *M1* равна: $d(M_1)$=3+4=7, длина маршрута *M2*: $d(M_2)$=5+1=6.

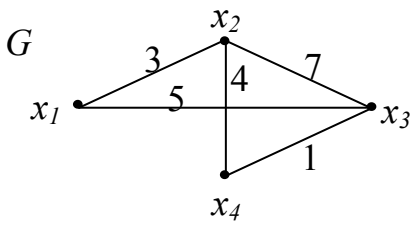


Рис. 4.3. **Нагруженный неорграф *G***

4.1.2. Основные понятия и определения для орграфа

Маршрут в орграфе, первая и последняя вершины которого различны, называется ***путем***. Примеры путей из вершины 1 в вершину 4 в орграфе *G* (рис. 4.4-*а*) выделены на рис. 4.4. *б – г.*

M1=1, 2, 1, 2, 3, 4; *M2*=1, 4; *M3*=1, 3, 4.

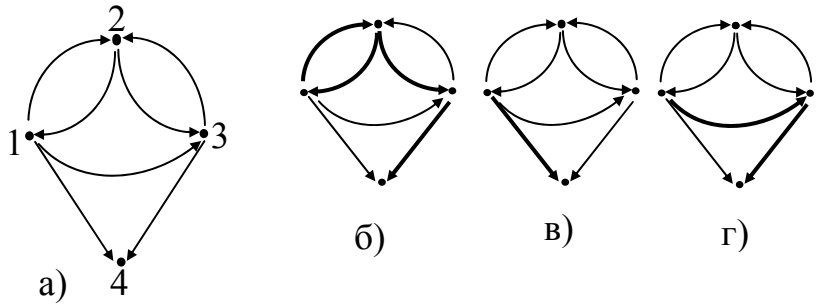


Рис. 4.4. **Пути в орграфах:**
а) орграф *G*; б) маршрут *M1*; в) маршрут *M2*; г) маршрут *M3*

Путь, в котором каждая вершина пройдена ровно один раз, называется ***простым*** путем.

Пути *M2* и *M3* - простые пути.

Путь, в котором равны начальная и конечная вершины называется ***контуром***.

Контуры *K1* = 1, 2, 1; *K2* = 2, 1, 2, 3, 2; *K3* = 1, 2, 3, 2, 1 в орграфе *G* (рис. 4.4-*а*) выделены на рис. 4.5 *а - в*.

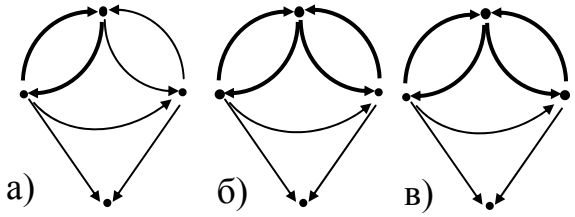


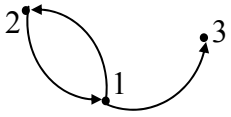
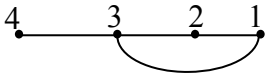
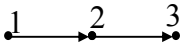
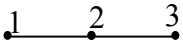
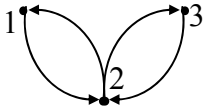
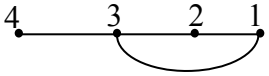
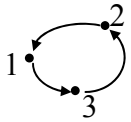
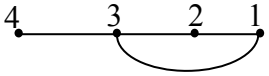
Рис. 4.5. **Контуры на орграфе *G:***
а) контур *K1*; б) контур *K2*; в) контур *K3*

Контур, образованный простым путем, называется ***простым*** контуром.
Контур *K1* - простой контур.

Различия в определениях между орграфами и неорграфами сведены в таблице 4.1. «Маршруты на графах. Основные определения»

Таблица 4.1

Маршруты на графах. Основные определения

Орграф	Неорграф
Путь *M1*= 1, 2, 1, 3 	Цепь *M1*= 1, 2, 1, 3,4 
Простой путь *M2* = 1, 2, 3 	Простая цепь *M2* = 1, 2, 3 
Контур *K1*= 1, 2, 3, 2, 1 	Цикл *C1* = 1, 2, 3, 1, 4, 3, 2, 1 
Простой контур *K1*= 1, 2, 3, 1 	Простой цикл *K1*= 1, 2, 3, 1 

4.2. Понятие достижимости и связности

4.2.1. Понятие достижимости и связности для неорграфа

В графе G вершина x считается ***достижимой*** из вершины y, если можно построить цепь между этими двумя вершинами. Вершина x считается достижимой сама из себя.

В графе G (рис. 4.6) вершина 2 достижима из вершин 6 и 7, вершина 1 достижима из вершины 4.

Множеством достижимых вершин для вершины x будем называть множество $R(x)=\{x_i\}$, $x_i \in X$, таких, что вершины x_i достижимы из x.

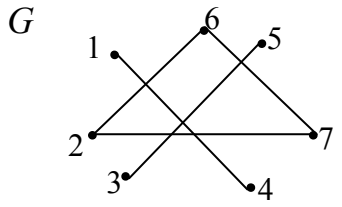


Рис. 4.6. **Достижимость на неорграфах**

Составим множества достижимости для вершин графа G (рис. 4.6).

$R(1)=R(4)=\{1,4\}$; $R(2)=R(6)=R(7)=\{2,6,7\}$; $R(3)=R(5)=\{3,5\}$.

Вершина y называется ***контрдостижимой*** из вершины x, если можно построить цепь из вершины x в вершину y. Заметим, что в неорграфе если вершина x достижима из вершины y, то y контрдостижима из x.

Множеством контрдостижимости $R^{-1}(x)=\{x_i\}$, $x_i \in X$ вершины x, называется множество вершин x_i, из которых x контрдостижима.

Очевидно, что для неорграфа $R(x)= R^{-1}(x)$, $\forall x \in X$.

Отношение достижимости на неорграфе является отношением эквивалентности. Отношение эквивалентности обладает свойствами рефлексивности, симметричности и транзитивности. Проверим выполнение этих свойств для отношения достижимости на неорграфе.

1) Рефлексивность. Действительно, $\forall x \in X$, где X множество вершин графа, выполняется условие: вершина x достижима сама из себя.

2) Симметричность. $\forall x, y \in X$, таких, что x достижима из y, следует что и y достижима из x.

3) Транзитивность, $\forall x, y, z \in X$, таких, что x достижима из y, y достижима из z, следует, что x достижима из z.

Множества достижимости графа G образуют классы эквивалентности отношения достижимости.

Неорграф называется ***связным***, если между любыми двумя его вершинами можно построить цепь.

Граф G (рис. 4.7) связный граф, граф G_1 (рис. 4.7) не является связным графом.

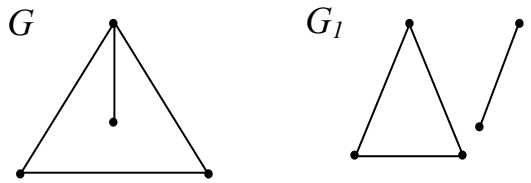


Рис. 4.7. **Связность графа**

Вершины, входящие в один класс эквивалентности отношения достижимости, образуют компоненту связности неорграфа.

Очевидно, что в связном графе одна компонента связности.

Граф, число компонент связности которого больше единицы, называется несвязным графом.

В графе G (рис. 4.6) три компоненты связности, в графе G (рис. 4.7) одна компонента связности, в графе G_1 (рис. 4.7) две компоненты связности.

Любая компонента связности есть связный подграф. Рассмотрим подграфы – компоненты связности графа G (рис. 4.6). Это подграфы G' (рис. 4.8) на вершинах 1, 4; G'' (рис. 4.8) на вершинах 2,6,7; G''' (рис. 4.8) на вершинах 5, 3.

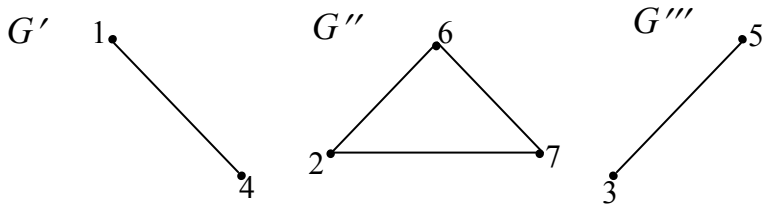


Рис. 4.8. **Компоненты связности графа *G***

4.2.2. Понятия достижимости и связности для орграфов

В орграфе вершина x считается ***достижимой*** из вершины y, если можно построить путь из вершины y в вершину x.

Для орграфов отношение достижимости не является отношением эквивалентности. Не выполняется свойство симметричности.

Действительно, в произвольном орграфе из существования пути из вершины y в вершину x не обязательно следует существование пути из вершины y в вершину x. Т.е., между двумя вершинами y и x, связанных путем, не обязательно существует путь с противоположным направлением.

В графе G (рис. 4.9) существует путь из вершины 1 в вершину 3, но нет пути из вершины 3 в вершину 1.

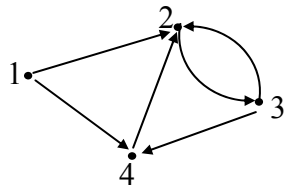


Рис. 4.9. **Орграф *G***

В орграфах вершина x считается ***контрдостижимой*** из вершины y, если можно построить путь из вершины y в вершину x.

Множество достижимости вершины x $R(x)=\{x_i\}$, $x_i \in X$, таких, что вершины x_i достижимы из вершины x.

Множество контрдостижимости вершины x $R^{-1}(x)=\{x_i\}$, $x_i \in X$, таких, что вершина x достижима из вершин x_i.

Множество достижимости для вершины 1 орграфа G (рис. 4.9) $R(1)=\{1,2,3,4\}$; множество контрдостижимости вершины 1 $R^{-1}=\{1\}$. Т.е. из вершины 1 можно построить путь в вершины 2, 3, 4, а в вершину 1 можно попасть из самой себя. Для орграфов дополнительно введено понятие взаимодостижимости.

Две вершины орграфа y и x называются взаимодостижимыми, если можно построить путь из вершины y в вершину x и путь из вершины x в вершину y.

В графе G (рис. 4.9) вершины 2 и 3 взаимодостижимы. Отношение взаимодостижимости на орграфах является отношением эквивалентности. Оно симметрично по определению. Рефлексивно, т.к. любая вершина орграфа считается достижимой сама из себя. Транзитивно, т.к. если x и y взаимодостижимы, и y и z взаимодостижимы, то взаимодостижимы и вершины x, z (рис. 4.10).

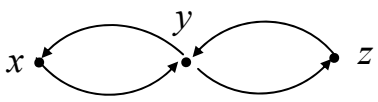


Рис. 4.10. **Транзитивность отношения взаимодостижимости**

Множества взаимодостижимости $W(x)=\{x_i\}$, $x_i \in X$ для каждой вершины графа $W(x)=\{x_i\}$, $x_i \in X$, где вершина x и вершины $x_i \in X$ взаимодостижимы, будут, таким образом, классами эквивалентности отношения взаимодостижимости.

Найдем множества взаимодостижимости для всех вершин графа G (рис. 4.9).

$$W(1)=\{1\};\ W(2)=\{2,3,4\}.$$

Классы эквивалентности отношения взаимодостижимости на орграфе называются ***сильными компонентами связности***.

В графе G (рис. 4.9) две сильные компоненты связности: подграф G', состоящий из одной вершины 1, и подграф G'' с вершинами {2,3,4} (рис. 4.11).

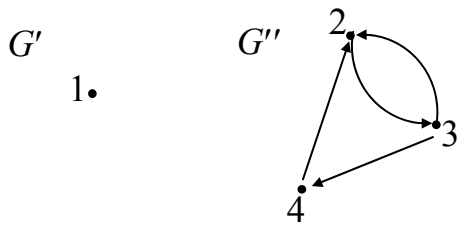


Рис. 4.11. **Сильные компоненты графа *G***

Орграф, с одной сильной компонентой называется ***сильно связным графом***.

Граф G (рис. 4.9) не является сильно связным орграфом, граф G_1 (рис. 4.12) – сильно связный орграф.

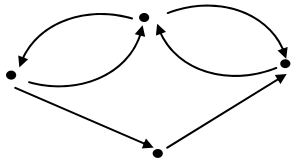


Рис. 4.12. **Сильно связный орграф G_1**

Перейдем от орграфа к неорграфу по следующей схеме:

1) удалим ориентацию ребер;
2) удалим петли;
3) оставим одно из кратных ребер.

Неорграф, полученный таким образом называется ***каркасом*** орграфа (рис. 4.13)

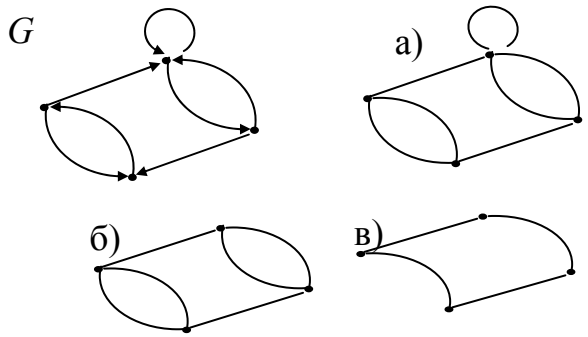


Рис. 4.13. **Построение каркаса орграфа *G*:**
а) удаление ориентации дуг; б) удаление петель;
в) удаление одного из кратных ребер

Если орграф не является сильно связным, а его каркас – связный неорграф, то орграф называется ***слабо связным***.

Граф *G* (рис. 4.9) – слабо связный граф.

Если каркас графа несвязный неорграф, то и орграф называется ***несвязным.***

4.3. Матрицы достижимости, контрдостижимости, взаимодостижимости

4.3.1. Булевы матрицы

Будем называть ***булевой матрицей*** такую матрицу, элементы которой равны либо нулю, либо единице.

Булевой матрицей можно считать, например, матрицу смежности графа.

Над булевыми переменными определены операции сложения - $\cup$ (***дизъюнкция***) и умножения - & (***конъюнкция***). Правила выполнения этих операций даны в таблице 4.2.

Таблица 4.2

Правила выполнения дизъюнкции и конъюнкции

Дизъюнкция	Конъюнкция
$0 \cup 0=0$	$0 \& 0=0$
$0 \cup 1=1$	$0 \& 1=0$
$1 \cup 0=1$	$1 \& 0=0$
$1 \cup 1=1$	$1 \& 1=1$

Пусть $A(n \times n)$ и $B(n \times n)$- булевы матрицы, тогда элементы матрицы $C(n \times n)=A(n \times n)+B(n \times n)$ определяются следующим образом:

$$c_{i,j} = a_{i,j} \cup b_{i,j}$$

Элементы матрицы $C(n \times n)=A(n \times n)*B(n \times n)$ определяются формулой:

$$c_{i,j}=a_{i,1} \& b_{1,i} \cup a_{i,2} \& b_{2,i} \cup \ldots \cup a_{i,n} \& b_{n,i}$$

Очевидно, что если при перемножении i-той строки и j-того столбца найдется хотя бы одна пара сомножителей (1,1), то элемент матрицы C $c_{i,j}$ будет равен единице. Запишите процедуры сложения и перемножения булевых матриц самостоятельно.

4.3.2. Матрицы достижимости, контрдостижимости и взаимодостижимости

Матрицей достижимости графа G называется матрица $R(n \times n)$, элементы которой вычисляются следующим образом:

$$r_{ij}=\begin{cases}1, & \text{если вершина } j \text{ достижима из вершины } i;\\ 0, & \text{в противном случае.}\end{cases}$$

Матрицей контрдостижимости графа G называется матрица $P(n \times n)$, элементы которой вычисляются следующим образом:

$$p_{ij}=\begin{cases}1, & \text{если вершина } i \text{ достижима из вершины } j;\\ 0, & \text{в противном случае.}\end{cases}$$

Отметим, что для связного неорграфа матрица достижимости состоит из одних только единиц. Для любых неорграфов матрицы контрдостижимости и достижимости равны. Это симметричные матрицы, элементы главных диагоналей которых - единичные элементы.

В общем случае матрицы достижимости и контрдостижимости орграфа не симметричные. Матрица контрдостижимости орграфа равна транспонированной матрице достижимости:

$$P=R^T$$

Матрицей взаимодостижимости графа G называется матрица $S(n\times n)$, элементы которой вычисляются по правилу:

$$s_{ij}=\begin{cases}1, & \text{если вершины } j \text{ и } i \text{ взаимодостижимы;}\\ 0, & \text{в противном случае.}\end{cases}$$

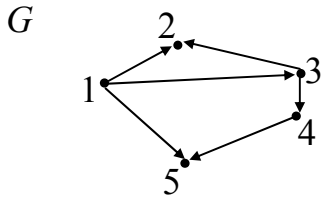
Очевидно, что для неорграфа матрицы достижимости и взаимодостижимости равны.

Элементы матрицы взаимодостижимости могут быть получены по следующему правилу:

$$c_{i,j}=a_{i,j}\ \&\ b_{i,j}.$$

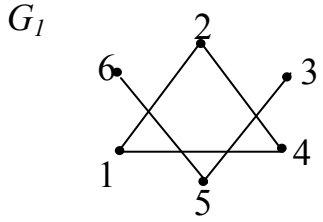
Отметим, что для сильно связного орграфа матрица взаимодостижимости состоит из одних единичных элементов.

Построим матрицы R, P и S для графов G и G_1 (рис. 4.15 и рис. 4.16).



$$R=\begin{bmatrix}1&1&1&1&1\\0&1&0&0&0\\0&1&1&1&1\\0&0&0&1&1\\0&0&0&0&1\end{bmatrix}\qquad P=R^T=\begin{bmatrix}1&0&0&0&0\\1&1&1&0&0\\1&0&1&0&0\\1&0&1&1&0\\1&0&1&1&1\end{bmatrix}\qquad S=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}$$

Рис. 4.14. **Орграф G и матрицы достижимости, контрдостижимости и взаимодостижимости**



$$R = P = S = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Рис. 4.15. **Неорграф G_1 и матрицы достижимости, контрдостижимости и взаимодостижимости**

4.3.3. Алгоритмы вычисления матрицы достижимости

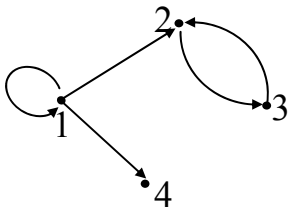
Очевидно, что матрицы контрдостижимости и взаимодостижимости получаются из матрицы достижимости путем несложных вычислений. Рассмотрим два способа получения матрицы достижимости.

Алгоритм построчного сложения.

Матрицу достижимости R можно получить, выполняя следующую последовательность действий.

В матрице смежности A рассматриваемого графа для всех элементов $a_{i,j}$, равных единице, выполнить булево сложение строк матрицы i и j, результат записать в строку i.

Вычислим матрицу достижимости R графа G_1 (рис. 4.16). Будем выделять просмотренные единичные элементы более жирным шрифтом.



$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Рис. 4.16. **Орграф G_1 и матрица смежности**

1) Отметим имеющиеся единицы на главной диагонали, как будто операция сложения уже была произведена, т.к. сложение строки с самой собой не изменит строку. Все появляющиеся далее единицы на главной диагонали будем отмечать подобным образом.

2) Элемент $a_{1,2} = 1$, сложим строки 1 и 2: $\begin{array}{cccc} & 1 & 1 & 0 & 1 \\ + & 0 & 1 & 0 & 0 \\ \hline & 1 & 1 & 1 & 1 \end{array}$.

Отметим все единицы в первой строке матрицы достижимости, т.к. с какой бы строкой не сложить единичную строку, она останется без изменения.

Матрица достижимости: $R = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

3) Выберем следующий неотмеченный единичный элемент матрицы R $a_{2,3}$. Выполним сложение строк 2 и 3: $\begin{array}{cccc} & 0 & 0 & 1 & 0 \\ + & 0 & 1 & 0 & 0 \\ \hline & 0 & 1 & 1 & 0 \end{array}$.

Матрица достижимости: $R = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

4) Выберем следующий неотмеченный единичный элемент матрицы R $a_{3,2}$. Выполним сложение строк 2 и 3: $\begin{array}{cccc} & 0 & 1 & 1 & 0 \\ + & 0 & 1 & 0 & 0 \\ \hline & 0 & 1 & 1 & 0 \end{array}$.

Матрица достижимости: $R = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

5) После выполнения последнего сложения в матрице R не осталось неотмеченных единиц. По условию достижимости любая вершина графа считается достижимой сама из себя, поэтому сложим матрицу R с единичной матрицей $E(n \times n)$:

$$E(n\times n)=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \qquad R = R + E = \begin{bmatrix}1&1&1&1\\0&1&1&0\\0&1&1&0\\0&0&0&1\end{bmatrix}.$$

R- матрица достижимости графа *G* (рис. 4.16).

Алгоритм булевого сложения матриц A^l, l изменяется от 1 до k, $k \le m$, m - число ребер рассматриваемого графа.

Второй способ получения матрицы достижимости *R* дает формула (4.1):

$$R=E\cup A^2\cup A^3\cup \ldots \qquad (4.1)$$

$A^l(n\times n)$ - матрица смежности графа в *l*-той степени, *E*- единичная матрица размерности $(n\times n)$. Вычисления проводятся до тех пор, пока при очередном выполнении дизъюнкции изменяется матрица *R*.

Продемонстрируем алгоритм для графа G_1 (рис. 4.16).
Матрица смежности графа G_1:

$$A = \begin{bmatrix}1&1&0&1\\0&0&1&0\\0&1&0&0\\0&0&0&0\end{bmatrix}.$$

На первом шаге вычислений матрица смежности *R* равна:

$$R^{(1)} = E + A = \begin{bmatrix}1&1&0&1\\0&1&1&0\\0&1&1&0\\0&0&0&1\end{bmatrix}.$$

Вычислим $A^2=A\&A$ и получим $R^{(2)}$ матрицу достижимости на втором шаге:

$$A^2 = \begin{bmatrix}1&1&1&1\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \qquad R^{(2)} = R^{(1)} + A^2 = \begin{bmatrix}1&1&1&1\\0&1&1&0\\0&1&1&0\\0&0&0&1\end{bmatrix}$$

Матрицы $R^{(1)}$ и $R^{(2)}$ не равны, следовательно, продолжим вычисления. Вычислим $A^3=A^2\&A$ и получим $R^{(3)}$, матрицу достижимости на третьем шаге:

$$A^3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R^{(3)} = R^{(2)} + A^3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матрицы $R^{(2)}$ и $R^{(3)}$ равны. Т.е., на третьем шаге вычисления в матрице достижимости не произошло ни каких изменений. Матрица $R^{(2)}$- искомая матрица достижимости R.

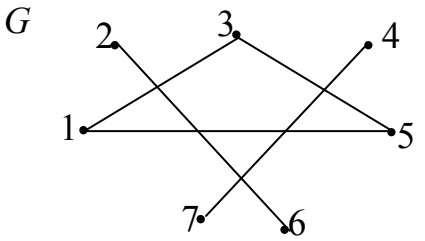
Для нахождения матрицы достижимости R графа G_1 (рис. 4.16) формула (4.1) выглядит следующим образом:

$$R=E\cup A\cup A^2.$$

Отметим, что максимальная степень матрицы смежности для конкретно взятого графа (для графа G (рис. 4.16) максимальная степень равна двум) равна длине максимальной простой цепи в неорграфе и пути в орграфе.

4.3.4. Выделение компонент связности

Опишем алгоритм выделения компонент связности неорграфа по матрице достижимости на примере графа G (рис. 4.17)



$$R = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Рис. 4.17. **Несвязный неорграф и его матрица достижимости**

В начале алгоритма число компонент связности полагается равным единице p:=1. Во множество S_p будем записывать вершины, входящие в компоненту связности с номером p.

Из матрицы R вычеркнем столбцы, в которых первый элемент единичный. Для графа G это столбцы 1, 3, 5.

Номера столбцов записываются во множество S_p. Для графа G S_1={1,3,5}. Первая компонента связности – подграф на вершинах 1, 3, 5. Затем из матрицы R вычеркнем строки с такими же номерами.

Матрица R после выполненных операций:

$$R' = \left[\begin{array}{c|cccc} x_i & \mathbf{2} & \mathbf{4} & \mathbf{6} & \mathbf{7} \\ \hline \mathbf{2} & 1 & 0 & 1 & 0 \\ \mathbf{4} & 0 & 1 & 0 & 1 \\ \mathbf{6} & 1 & 0 & 1 & 0 \\ \mathbf{7} & 0 & 1 & 0 & 1 \end{array}\right]$$

Если не вычеркнуты все столбцы из матрицы R, то число компонент связности увеличивается на единицу $p:=p+1$.

Вычеркнем из полученной матрицы столбцы и строки с номером n_1, $n_2,\ldots, n_k$, если первый элемент в столбце $n_1, n_2,\ldots, n_k$ – единица. Для графа G $S_2=\{2,6\}$.

Вторая компонента связности – подграф на вершинах 2, 6.

Матрица R после выполненных операций:

$$R'' = \left[\begin{array}{c|cc} x_i & \mathbf{4} & \mathbf{7} \\ \hline \mathbf{4} & 1 & 1 \\ \mathbf{7} & 1 & 1 \end{array}\right].$$

На третьем шаге выполняем действия, описанные ранее с измененной матрицей достижимости R''. Для графа G $S_3=\{4,7\}$. Третья компонента связности – подграф на вершинах 4, 7.

Для графа G после вычеркивания столбцов 4 и 7 алгоритм заканчивает свою работу.

Для произвольного графа алгоритм заканчивает работу, если в матрице достижимости вычеркнуты все строки и столбцы.

Для графа G число подсчитанных компонент связности равно *трем*.

Матрицы смежности подграфов, соответствующих компонентам связности выделяются из матрицы смежности графа (рис. 4.18).

$$A(G) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad A(G) = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Рис. 4.18. **Выделение матрицы смежности компонент связности**

В графе G первая компонента связности - подграф G_1 на вершинах из $S_1=\{1,3,5\}$. Выделим столбцы и строки с номерами 1, 3, 5. Элементы, находящиеся на пересечении столбцов и строк, составят матрицу смежности для подграфа G_1.

$$A(G_1)=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 0\end{bmatrix}.$$

Замечание. Для выделения компонент сильной связности орграфа используется тот же самый алгоритм, но на матрице взаимодостижимости.

4.4. Метрики связного неорграфа

Минимальной цепью между вершинами x и y в неорграфе G называется цепь с минимальным количеством ребер.

Минимальным путем между вершинами x и y в орграфе G называется путь с минимальным количеством дуг.

Теорема 4.1 *(о существовании минимальной цепи).* Пусть дан граф G. Если в графе G существует цепь между вершинами x и y, то существует и минимальная цепь между вершинами x и y.

Доказательство. Пусть в графе G между вершинами x и y существуют несколько цепей $C_1, C_2 \ldots C_n$ с длинами $d_1, d_2 \ldots d_n$ соответственно. Среди чисел $d_1, d_2 \ldots d_n$ всегда можно найти минимальное число. Найденное минимальное d_i будет длиной минимальной цепи c_i.

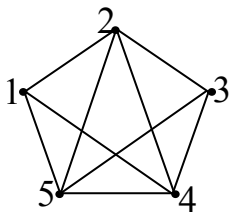
Для орграфов существование минимального пути доказывается аналогично.

Обозначим минимальную цепь из вершины x в вершину y как $d(x,y)$. Так же определим, что если цепь из вершины x в вершину y не существует, то $d(v,w)=\infty$. Будем считать, что $d(x,x)=0$.

Матрицей минимальных расстояний $D(n\times n)$ графа $G(X,V)$ называется матрица, элементы которой определяются по правилу:

$$d_{i,j}=\begin{cases}0, \text{если } i=j\\ \infty, \text{если цепь между вершинами } i \text{ и } j \text{ не } \exists\\ d(i,j), \text{в противном случае}\end{cases}$$

Построим матрицу минимальных расстояний для графа G (рис. 4. 19).



$$D(G)=\begin{bmatrix} 0 & 1 & 2 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Рис. 4.19. **Граф *G* и матрица минимальных расстояний**

Для графа $G=(X,V)$ будем называть ***диаметром*** графа G величину $d(G)=\max\limits_{x,y\in X} d(x,y)$ (максимальное из всех минимальных расстояний).

Пусть x – произвольная вершина из X. Величина $r(x)=\max\limits_{y\in X} d(x,y)$ будет называться ***максимальным удалением (эксцентриситетом)*** в графе G от вершины x (максимальное из всех минимальных расстояний для вершины x).

Радиусом графа G называется величина $r(G)=\min\limits_{x\in X} r(x)$ (минимальный из эксцентриситетов всех вершин).

Любая вершина графа, такая, что $r(x)=r(G)$ называется ***центром*** графа.

Найдем диаметр, радиус, центры графа G (рис. 4.19). Используем для вычислений матрицу минимальных расстояний графа.

Найдем максимальные элементы каждой строки и запишем их в вектор F (рис. 4.20):

$$D(G)=\left[\begin{array}{ccccc|c} 0 & 1 & 2 & 1 & 1 & \mathbf{2} \\ 1 & 0 & 1 & 1 & 1 & \mathbf{1} \\ 2 & 1 & 0 & 1 & 1 & \mathbf{2} \\ 1 & 1 & 1 & 0 & 1 & \mathbf{1} \\ 1 & 1 & 1 & 1 & 0 & \mathbf{1} \\ \hline & & & & & F \end{array}\right]$$

Рис. 4.20. **Нахождение метрик графа *G***

По определению диаметра и способу построения матрицы D, $d(G)$ - это максимальный элемент из всех элементов матрицы D. По способу построения вектора F, этот элемент обязательно попадет в F. Тогда диаметр $d(G) = \max\limits_{i=\overline{1,n}} (f_i)$. Для графа G (рис. 4.23) $d(G)=2$.

По способу построения вектора F и из определения эксцентриситета вершины выполняется равенство $r(i)=f_i$. Тогда для графа G (рис. 4.20) эксцентриситеты вершин равны:

$$r(1)=2;\ r(2)=1;\ r(3)=2;\ r(4)=1;\ r(5)=1.$$

Тогда радиусом графа будет минимальный элемент вектора F. Для графа G (рис. 4.20) $r(G)=1$.

По определению центра графа и способу построения вектора F, если $f_i=r(G)$, то вершина i графа есть центр графа. В графе G (рис. 4.20) три центра - вершины 1,4 и 5.

4.5. Обход графа

Иногда в задачах, связанных с теорией графов, требуется найти цепь между двумя вершинами. К этой задаче сводится, например, поиск выхода из лабиринта, если проходы лабиринта считать ребрами графа, а повороты и тупики вершинами. Либо требуется каким-то образом отметить все вершины графа или все ребра графа. Например, отметить все рабочие линии электросети, считая, что по нерабочим линиям проход невозможен. Остановимся на задаче разметки всех вершин и ребер графа, как на более общей задаче. Решим эту задачу с помощью алгоритма обхода графа «в глубину». Далее будет рассмотрен и алгоритм обхода графа «в ширину».

Алгоритм получил свое название из-за того, что из заданной вершины графа стремятся получить как можно более длинную простую цепь, т.е. пройти как можно дальше «вглубь» (рис. 4.21.)

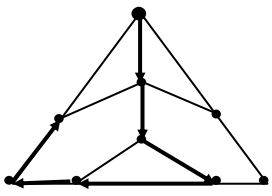


Рис. 4.21. **Понятие продвижения «вглубь» на графе**

Для работы алгоритма требуется хранить последовательность переходов от одной вершины графа к другой. Используем для этого стек.

Стек – это связная структура хранения данных, добавление и удаление элементов которого, происходит по правилу ***«последний вошел, первый вышел»***. В литературе стек описывают аббревиатурой LIFO. Чтобы нагляднее представить себе стек, вспомните детскую пирамидку: снять можно только то кольцо, которое было надето последним.

Рассмотрим работу алгоритма на примере графа G (рис. 4.22). Будем выполнять общий случай алгоритма обхода «в глубину» - требуется обойти все ребра и все вершины исследуемого графа. По мере прохождения будем отмечать все вершины и ребра различными метками 1,2, … n.

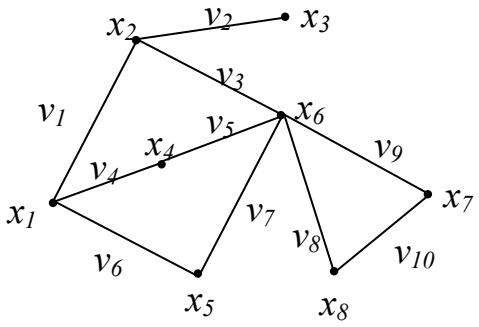


Рис. 4.22. **Неорграф *G***

Начнем обход графа с вершины x_1 (вершина выбирается произвольно). Поставим вершине x_1 метку 1, всем остальным вершинам поставим метки 0. Ребрам так же поставим метки 0. Вершину x_1 запишем в стек, который в начале работы алгоритма был пустым. В начале работы алгоритма ПУТЬ = $\varnothing$. Вершину x_1 запишем в множество ПУТЬ. В дальнейшем, в ПУТЬ будем записывать все проходимые вершины. Состояние меток вершин, меток ребер, стека и множества ПУТЬ представлено на рис. 4.23-*а*. *MV* - массив меток ребер.

Пройденные ребра на рис. 4.23 выделены более жирными линиями, а текущая вершина – большей точкой.

Выберем произвольно вершину, смежную текущей вершине x_1, такую, что ее метка равна нулю. Перейдем к выбранной вершине (сделаем ее текущей). Пусть это будет вершина x_2. Поставим ей метку 2, ребру, по которому переходили в вершину x_2, поставим метку 1. Запишем вершину x_2 в стек и в ПУТЬ (рис. 4.23-*б*).

Далее на рис. 4.23(*в*) в графе строится цепь $x_1v_1x_2v_3x_3$, по тем же правилам, что и построение пути из вершины x_1 в вершину x_2.

Для текущей на этом шаге алгоритма вершины x_3 можно рассмотреть одну смежную вершину x_2, которая проходилась ранее. Для вершины x_3 отмечены все смежные вершины и инцидентные ребра. Удалим вершину x_3 из стека и вернемся к вершине x_2 (рис. 4.23- *г*). Из вершины x_2 строится путь $x_6 v_5 x_4$ (рис. 4.23-*д,е*). На следующем шаге необходимо найти вершину, смежную текущей вершине x_4. Вершина x_1 ранее проходилась, но ребро (x_4, x_1) не отмечалось. Отметим это ребро, стек не изменяется, в ПУТЬ записывается последовательность $x_1 x_4$. При дальнейшем обходе графа для вершины x_4 не нашлось неотмеченных смежных вершин и инцидентных ребер, вершина удаляется из стека, текущей становится вершина x_6 (рис. 4.23-*ж*). Аналогичные действия отражены на рис. 4.23-*з,и.* Далее строится путь $x_6 v_9 x_7 v_{10}$. Алгоритм закончит свою работу после того, как будут отмечены все ребра и вершины (рис. 4.23-*м*). В результате работы алгоритма в массиве ПУТЬ будут сохранены вершины, в порядке их прохождения по графу.

Запишем алгоритм обхода графа «в глубину». В начале работы алгоритма СТЕК пуст.

a – произвольная вершина из множества вершин графа.
$nv = 1$ – номер текущей вершины.
$nr = 0$ – номер текущего ребра.
$mv[x]$ – метка вершины x.
$mr[(x, y)]$ – метка ребра между вершинами x и y.
n – количество вершин.
m – количество ребер.
X – множество вершин.
V – множество ребер.

ЦИКЛ ДЛЯ $\forall v \in V$
  $mr(v) = 0$
ВСЕ ЦИКЛ
ЦИКЛ ДЛЯ $\forall x \in x$
  $mv(x) = 0$
ВСЕ ЦИКЛ
$a \rightarrow$ СТЕК
$mv[a] := nv;\ nv := nv + 1$
ПОКА($nv < n$ ИЛИ $nr < m$)
  ПОКА ($\exists$ вершина x, смежная с a и ($mv[x] = 0$ ИЛИ $mr[(a, x)] = 0$)

ЕСЛИ ($mv[x] = 0$)

ТО $mv[x] := nv$

$mr[(a, x)] := nr$

$nv := nv + 1$

$nr := nr + 1$

$x \to a$

$a \to$ СТЕК

$a \to$ ПУТЬ

ИНАЧЕ $mr[(a, x)] := nr$

$nr := nr + 1$

ВСЕ ЕСЛИ

ВСЕ ПОКА

Прочитать из СТЕК a {не удаляя a из СТЕК}

ЕСЛИ (($\forall$ x, смежная с a) И ($mv[x] \neq 0$) И ($mr[(a, x)] \neq 0$))

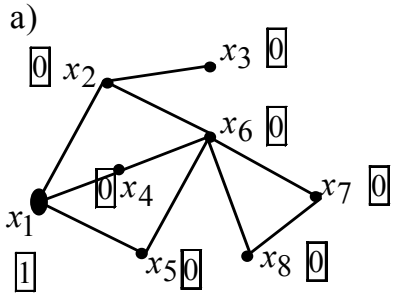
ТО СТЕК $\to a$ *{удалить вершину a из стека}*

ИНАЧЕ $a \to$ ПУТЬ

ВСЕ ЕСЛИ

ВСЕ ПОКА

а)

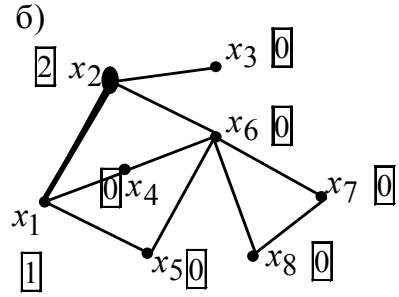


Стек: x_1

v_i, i	mv_i	v_i, i	mv_i
1	0	6	0
2	0	7	0
3	0	8	0
4	0	9	0
5	0	10	0

ПУТЬ $[x_1]$

б)

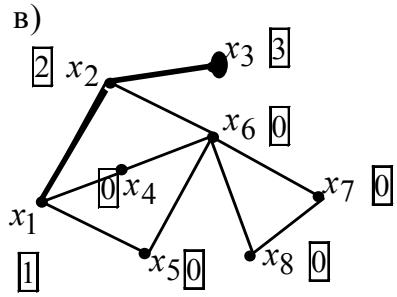


Стек: x_2, x_1

v_i, i	mv_i	v_i, i	mv_i
1	1	6	0
2	0	7	0
3	0	8	0
4	0	9	0
5	0	10	0

ПУТЬ $[x_1x_2]$

в)

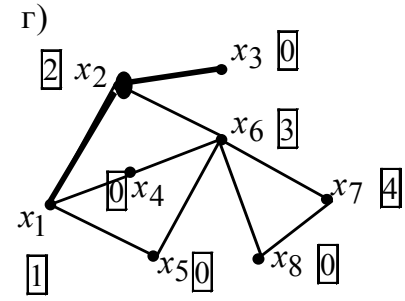


Стек: x_3, x_2, x_1

v_i, i	mv_i	v_i, i	mv_i
1	1	6	0
2	2	7	0
3	0	8	0
4	0	9	0
5	0	10	0

ПУТЬ $[x_1x_2x_3]$

г)

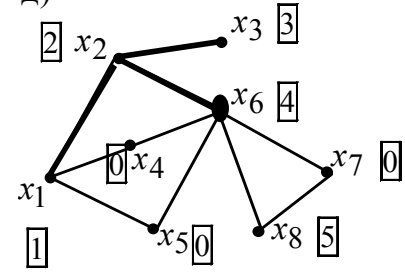


Стек: x_2, x_1

v_i, i	mv_i	v_i, i	mv_i
1	1	6	0
2	2	7	0
3	0	8	0
4	0	9	0
5	0	10	0

ПУТЬ $[x_1x_2x_3x_2]$

д)



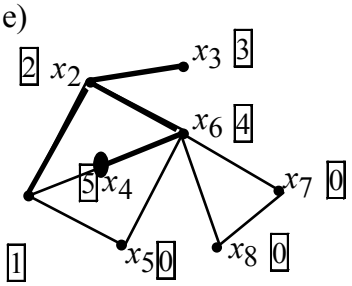
Стек: x_6, x_2, x_1

v_i, i	mv_i	v_i, i	mv_i
1	1	6	0
2	2	7	0
3	3	8	0
4	0	9	0
5	0	10	0

ПУТЬ $[x_1x_2x_3x_2x_6]$

Рис. 4.23. Обход графа «в глубину»

е)



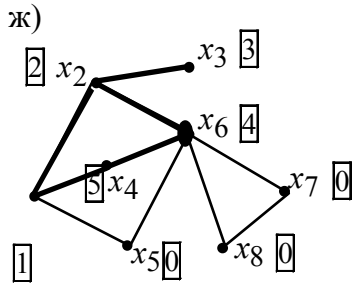
Стек: x_4, x_6, x_2, x_1

v_i, i	mv_i
1	1
2	2
3	3
4	0
5	4

v_i, i	mv_i
6	0
7	0
8	0
9	0
10	0

ПУТЬ $[x_1 x_2 x_3 x_2 x_6 x_4]$

ж)



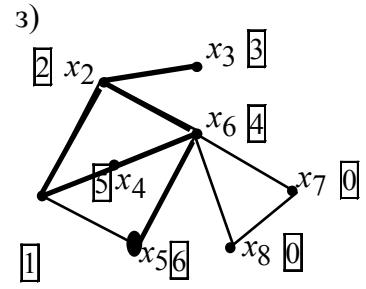
Стек: x_6, x_2, x_1

v_i, i	mv_i
1	1
2	2
3	3
4	5
5	4

v_i, i	mv_i
6	0
7	0
8	0
9	0
10	0

ПУТЬ $[x_1 x_2 x_3 x_2 x_6 x_4 x_1 x_4 x_6]$

з)



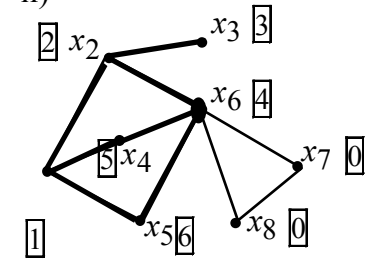
Стек: x_5, x_6, x_2, x_1

v_i, i	mv_i
1	1
2	2
3	3
4	5
5	4

v_i, i	mv_i
6	0
7	6
8	0
9	0
10	0

ПУТЬ $[x_1 x_2 x_3 x_2 x_6 x_4 x_1 x_4 x_6 x_5]$

и)



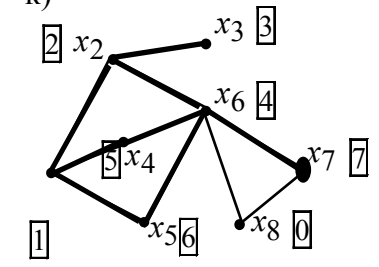
Стек: x_6, x_2, x_1

v_i, i	mv_i
1	1
2	2
3	3
4	5
5	4

v_i, i	mv_i
6	7
7	6
8	0
9	0
10	0

ПУТЬ $[x_1 x_2 x_3 x_2 x_6 x_4 x_1 x_4 x_6 x_5 x_1 x_5 x_6]$

к)



Стек: x_7, x_6, x_2, x_1

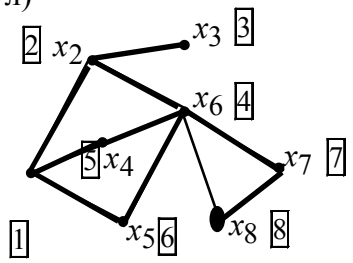
v_i, i	mv_i
1	1
2	2
3	3
4	5
5	4

v_i, i	mv_i
6	7
7	6
8	0
9	8
10	0

ПУТЬ $[x_1 x_2 x_3 x_2 x_6 x_4 x_1 x_4 x_6 x_5 x_1 x_5 x_6 x_7]$

Продолжение рис. 4.23. Обход графа «в глубину»

л)



Стек

x_8
x_7
x_6
x_2
x_1

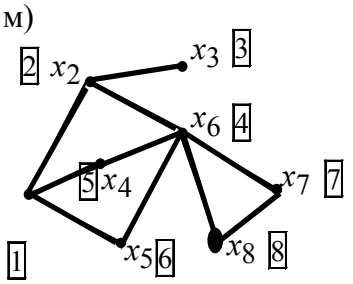
v_i, i	mv_i
1	1
2	2
3	3
4	5
5	4

v_i, i	mv_i
6	7
7	6
8	0
9	8
10	9

ПУТЬ

$[x_1x_2x_3$
$x_2x_6x_4$
$x_1x_4x_6$
$x_5x_1x_5$
$x_6x_7x_8]$

м)



Стек

x_8
x_7
x_6
x_2
x_1

v_i, i	mv_i
1	1
2	2
3	3
4	5
5	4

v_i, i	mv_i
6	7
7	6
8	10
9	8
10	9

ПУТЬ

$[x_1x_2x_3$
$x_2x_6x_4$
$x_1x_4x_6$
$x_5x_1x_5$
$x_6x_7x_8$
$x_6x_8]$

Продолжение рис. 4.23. Обход графа «в глубину»

Алгоритм обхода «в глубину» можно использовать для выделения компонент связности. Рассмотрим применение алгоритма для графа *G* (рис. 4.24).

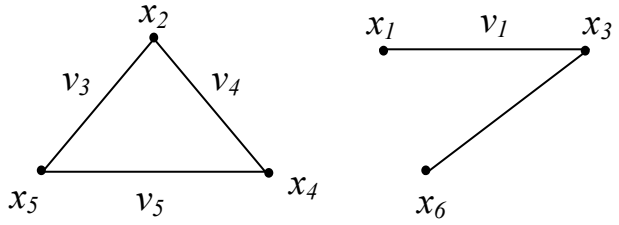


Рис. 4.24**. Несвязный граф *G***

Выберем произвольно начальную вершину, пусть это будет x_1, обнулим метки всех вершин и ребер, очистим стек.

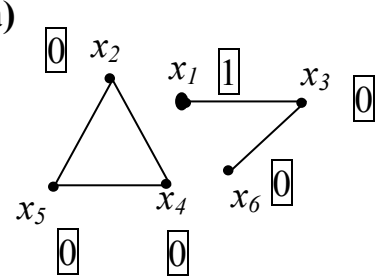
Число компонент связности на начало работы алгоритма p=1.

Будем записывать во множество S_p вершины графа *G*, которые принадлежат p-той компоненте связности (рис. 4.25-*а*).

Далее был использован алгоритм обхода в «глубину» (рис. 4.25 *а-е*). При не полностью пройденном графе (не проходились вершины x_2, x_5, x_4 и ребра v_3, v_4, v_5 на рис. 4.25–*е*), стек стал пустым.

Таким образом, в процессе обхода выделился максимальный связный подграф, содержащий начальную вершину обхода, а это есть компонента связности графа *G*.

а)



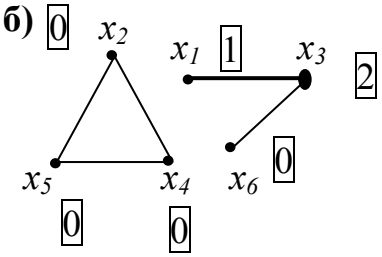
Стек

x_1

v_i, i	mv_i
1	0
2	0
3	0
4	0
5	0

p=1
S_p={x_1}

б)



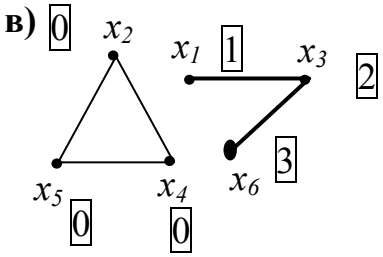
Стек

x_3
x_1

v_i, i	mv_i
1	1
2	0
3	0
4	0
5	0

p=1
S_p={x_1 x_3}

в)



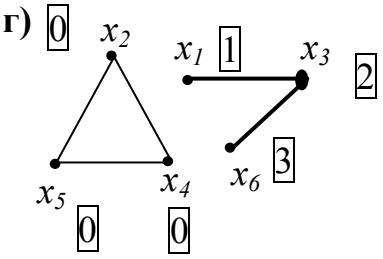
Стек

x_6
x_3
x_1

v_i, i	mv_i
1	1
2	1
3	0
4	0
5	0

p=1
S_p={x_1 x_3 x_6}

г)



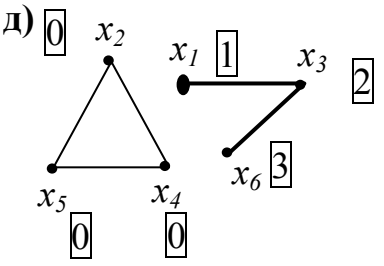
Стек

x_3
x_1

v_i, i	mv_i
1	1
2	1
3	0
4	0
5	0

p=1
S_p={x_1 x_3 x_6}

д)



Стек

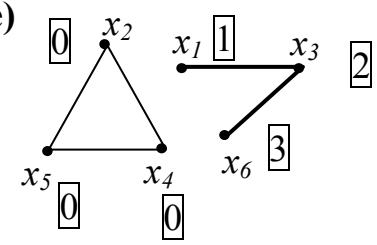
x_1

v_i, i	mv_i
1	1
2	1
3	0
4	0
5	0

x_1

Рис. 4.25. **Выделение компонент связности**

е)

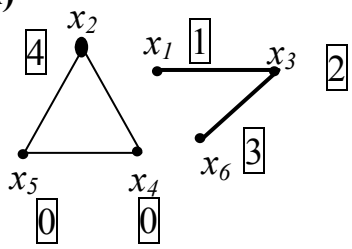


Стек

v_i, i	mv_i
1	1
2	1
3	0
4	0
5	0

p=1
S_p={x_1 x_3 x_6}

ж)



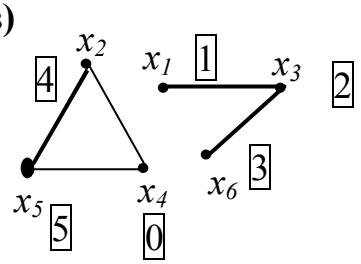
Стек

x_2

v_i, i	mv_i
1	1
2	1
3	0
4	0
5	0

p=2
S_1={x_1 x_3 x_6}
S_p={x_2}

з)



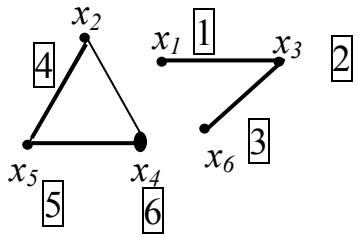
Стек

x_5
x_2

v_i, i	mv_i
1	1
2	1
3	1
4	0
5	0

p=2
S_1={x_1 x_3 x_6}
S_p={x_2 x_5}

и)



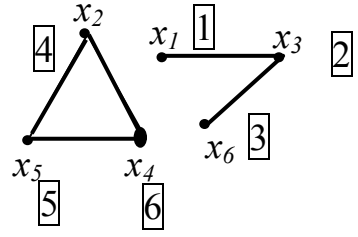
Стек

x_4
x_5
x_2

v_i, i	mv_i
1	1
2	1
3	1
4	0
5	1

p=2
S_1={x_1 x_3 x_6}
S_p={x_2 x_5 x_6}

к)



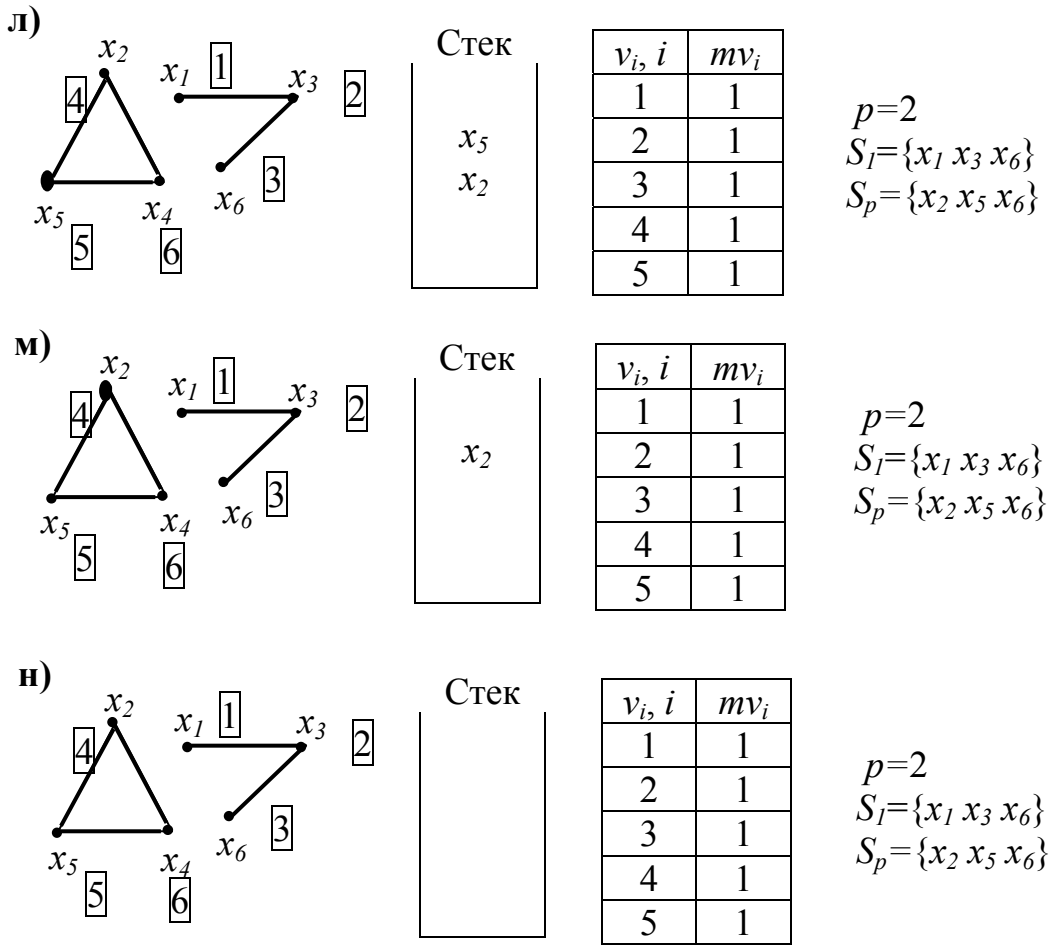
Стек

x_4
x_5
x_2

v_i, i	mv_i
1	1
2	1
3	1
4	1
5	1

p=2
S_1={x_1 x_3 x_6}
S_p={x_2 x_5 x_6}

Продолжение рис. 4.25. **Выделение компонент связности**



Продолжение рис. 4.25. **Выделение компонент связности**

Будем увеличивать число компонент связности на единицу p:=p+1 всякий раз, как в процессе работы стек становится пустым, а в графе G есть не пройденные вершины.

За текущую вершину в таком случае выбирается любая вершина, метка которой равна нулю (рис. 4.25 *ж-н*).

Алгоритм работает, пока не пройдены все вершины графа.

Запишите самостоятельно алгоритм выделения компонент связности на псевдокоде.

4.6. Минимальные маршруты на связных неорграфах

Ранее уже давалось определение минимального пути из вершины x в вершину y.

Теорема 4.2. Минимальный маршрут на графе всегда есть простая цепь.

Доказательство. Пусть существует минимальный маршрут из вершины x в вершину y, не являющийся простой цепью M=$\boldsymbol{x}$ x_1 x_2...x_k $\boldsymbol{y}$. Внутри

маршрута вершины пронумерованы по порядку. Длина маршрута M равна k+1. Тогда найдутся номера i и j, такие, что $1 \le i < j \le k$ и $x_i = x_j$. Рассмотрим маршрут $M_1 = \boldsymbol{x}\ x_1\ x_2 \ldots x_i\ x_{i+1}\ x_k\ \boldsymbol{y}$. Его длина меньше, чем длина маршрута M. Следовательно, маршрут M не минимальный.

Рассмотрим случай, когда $x_i = x$. Тогда маршрут $M_2 = \boldsymbol{x}\ x_{i+1} \ldots x_k\ \boldsymbol{y}$ имеет длину меньшую, чем длина маршрута M.

Рассмотрим случай, когда $x_i = y$. Тогда маршрут $M_3 = \boldsymbol{x}\ x_1\ x_2 \ldots x_{i-1}\ \boldsymbol{y}$ имеет длину меньшую, чем длина маршрута M.

Следовательно, минимальный маршрут не может быть не простой цепью.

Рассмотренный далее волновой алгоритм находит минимальный маршрут между двумя заданными вершинами x и y. Алгоритм получил свое название из-за того, что от заданной начальной вершины как бы распространяется «волна». «Волна» первого уровня достигнет всех вершин, в которые можно попасть, пройдя по одному ребру, «волна» второго уровня достигнет всех вершин, в которые можно попасть, пройдя ровно по двум ребрам и т.д.

Найдем минимальную цепь между вершинами x_1 и x_5 в графе G (рис. 4.26).

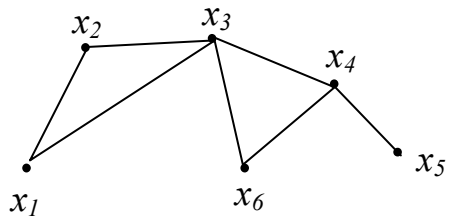


Рис. 4.26. **Связный неорграф *G***

Поставим начальной вершине пути x_1 метку, равную нулю, всем остальным вершинам графа метки, равные $\boxed{-1}$ (рис. 4.27-*а*). Множество вершин с метками, равными k, будем называть фронтом волны k-того уровня. Во фронт волны нулевого уровня вошла вершина x_1.

Пусть переменная k=0. Найдем вершины, смежные вершинам из фронта волны k-того уровня. Если их метки меньше нуля, то поставим им метки, равные k+1. Эти вершины образуют фронт волны k+1-го уровня (рис. 4.27-*б*). Метками $\boxed{1}$ были отмечены вершины x_2 и x_3. Конечная вершина пути x_5 не была отмечена на этом шаге.

Будем выполнять далее следующие действия. Увеличим значение переменной k на единицу. Будем отмечать вершины, смежные вершинам из фронта волны k-го уровня, в том случае, если их метки меньше нуля (рис. 4.27-*в*). Для вершины x_2 смежные вершины x_1 и x_3. Но их метки не меньше нуля, поэтому они не изменяются. Для вершины x_3 нашлись две вершины, с метками $\boxed{-1}$, вершины x_6 и x_4. Поставим им метки, равные k+1. Т.к. конечная вершина

не отмечена, то увеличим значение k на единицу и будем продолжать выполнять те же самые действия (рис.4.27-*г*). На этом шаге была достигнута конечная вершина маршрута. Первая часть алгоритма – прямая, заканчивает свою работу. Длина маршрута равна k+1. Для графа G длина маршрута равна трем.

После этого запишем непосредственно сам маршрут, пройдя от вершины x_5 до вершины x_1. Эта часть алгоритма называется обратным обходом.

Будем записывать вершины в порядке их прохождения в массив ПУТЬ. Элементу массива ПУТЬ[k+1] присвоим значение текущей на этом шаге вершины x_5 (рис. 4.27-*д*). Пока не достигнута начальная вершина пути x_1 будем выполнять следующие действия. Найдем вершину, смежную текущей и имеющей метку, равную k. Найденную вершину сделаем текущей и уменьшим значение k на единицу (рис. 4.27 *е-з*).

Запишем волновой алгоритм на псевдокоде. Всем вершинам графа присвоим метки –1.

```
ЦИКЛ (для всех x ∈ X)
        MV(x):=-1;
ВСЕ_ЦИКЛ
```

Отметим начальную вершину пути a меткой 0, переменной k присвоим значение k:=0. Запишем начальную вершину во фронт волны k-го уровня:

FW_k:={a}.

$MV(a)$:=0; k:=0; FW_k:={a};

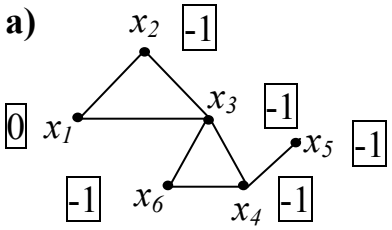
Для того, чтобы разметить все вершины графа, организуем цикл, который будет выполняться до тех пор, пока не будет найдена конечная вершина пути или переменная k не станет больше m, числа ребер:

ЦИКЛ_ПОКА ($b \notin FW_k$) ИЛИ ($k \le m$)

Найдем все вершины, смежные вершинам из фронта волны k-того уровня, метки которых меньше нуля. Запишем их во фронт волны k+1-го уровня и поставим им метки, равные k+1.

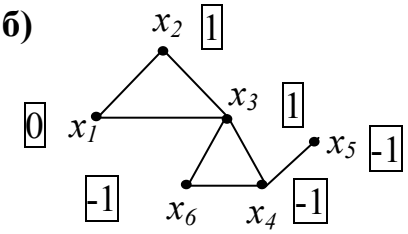
```
FW_k:=∅;
ЦИКЛ (для  x ∈ FW_k)
  ЦИКЛ (для всех y ∈ X)
    ЕСЛИ СОСЕД (x,y) И MV(y)=-1
         ТО MV(y):=k+1
              y ∪ FW_{k+1};
    ВСЕ_ЕСЛИ
  ВСЕ_ЦИКЛ
 ВСЕ_ЦИКЛ
```

а)



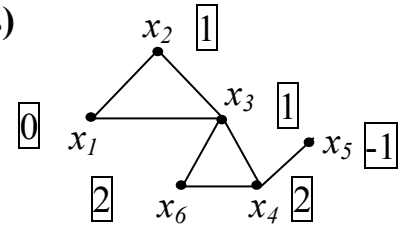
$FW_0=\{x_1\}$

б)



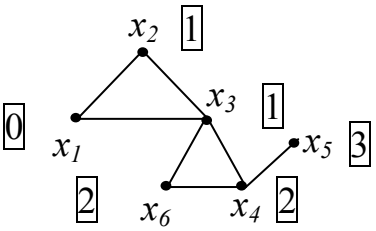
$FW_0=\{x_1\}$
$FW_1=\{x_3, x_2\}$

в)



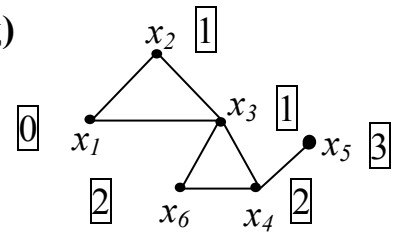
$FW_0=\{x_1\}$
$FW_1=\{x_3, x_2\}$
$FW_2=\{x_6, x_4\}$

г)



$FW_0=\{x_1\}$
$FW_1=\{x_3, x_2\}$
$FW_2=\{x_6, x_4\}$
$FW_3=\{x_5\}$

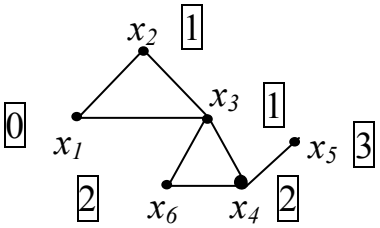
д)



$FW_0=\{x_1\}$
$FW_1=\{x_3, x_2\}$
$FW_2=\{x_6, x_4\}$
$FW_3=\{x_5\}$

ПУТЬ[3]:=x_5

е)

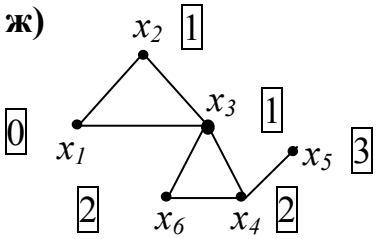


$FW_0=\{x_1\}$
$FW_1=\{x_3, x_2\}$
$FW_2=\{x_6, x_4\}$
$FW_3=\{x_5\}$

ПУТЬ[3]:=x_5
ПУТЬ[2]:=x_4

Рис. 4.27. **Нахождение минимального пути**

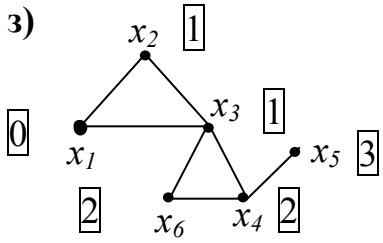
ж)



$FW_0=\{x_1\}$
$FW_1=\{x_3, x_2\}$
$FW_2=\{x_6, x_4\}$
$FW_3=\{x_5\}$

ПУТЬ[3]:=x_5
ПУТЬ[2]:=x_4
ПУТЬ[1]:=x_3

з)



$FW_0=\{x_1\}$
$FW_1=\{x_3, x_2\}$
$FW_2=\{x_6, x_4\}$
$FW_3=\{x_5\}$

ПУТЬ[3]:=x_5
ПУТЬ[2]:=x_4
ПУТЬ[1]:=x_3
ПУТЬ[0]:=x_1

Продолжение рис. 4.27. **Нахождение минимального пути**

Увеличим значение k на единицу.

k:=k+1;

Закончим прямой ход алгоритма.

ВСЕ_ЦИКЛ

Если конечная вершина пути $b \in FW_k$, то будем выполнять обратный обход (построение пути), в противном случае необходимо выдать сообщение о том, что пути из a в b не существует.

ЕСЛИ ($b \in FW_k$) ТО …
ИНАЧЕ Печать («Пути из a в b не существует.»)
ВСЕ_ЕСЛИ

Запишем алгоритм обратного обхода. Организуем цикл, который выполняется пока не будет достигнута начальная вершина пути a. Обозначим за z текущую вершину, изначально равную b.

z:=b; Печать («Длина пути равна», k, «Путь: », z);
ЦИКЛ_ПОКА ($z \neq a$)
  ЦИКЛ (для всех вершин $x \in X$)
    ЕСЛИ СОСЕД (x, z) И (MV(x)=k-1)
      ТО z:=x;
        k:=k-1;
        Печать(z);
    ВСЕ_ЕСЛИ
  ВСЕ_ЦИКЛ
ВСЕ_ЦИКЛ

На рис. 4.28 волновой алгоритм представлен полностью.

Если граф нагруженный, т.е. каждое ребро имеет свой вес, то задача нахождения минимального маршрута в таком графе сводится к нахождению маршрута, сумма ребер которого минимальна.

В графе G (рис. 4.26) минимальный маршрут из вершины x_1 в вершину x_2 - $M=x_1 x_3 x_2$. Сумма весов ребер этого маршрута равна трем.

Рассмотрим два алгоритма поиска кратчайших путей на нагруженных графах.

```
НАЧАЛО
ЦИКЛ (для всех  x ∈ X)
        MV(x):=-1;
ВСЕ_ЦИКЛ
MV(a):=0;  k:=0;  FW_k:={a};

ЦИКЛ_ПОКА (b ∉ FW_k) ИЛИ (k ≤ m)
          FW_k:=∅;
ЦИКЛ (для всех x ∈ FW_k)
  ЦИКЛ (для всех y ∈ X)
    ЕСЛИ СОСЕД (x,y) И MV(y):=-1,
        ТО MV(y):=k+1;
             y ∪ FW_{k+1};
    ВСЕ_ЕСЛИ
  ВСЕ_ЦИКЛ
 ВСЕ_ЦИКЛ
 k:=k+1;
ВСЕ_ЦИКЛ
ЕСЛИ (b∈ FW_k) ТО (обратный обход)
            z:=b; Печать («Длина пути равна», k, «Путь: », z);
            ЦИКЛ_ПОКА (z ≠ a)
              ЦИКЛ (для всех вершин x ∈ X)
                ЕСЛИ СОСЕД (x, z) И (MV(x):=k-1)
                      ТО z:=x;
                         k:=k-1;
                        Печать(z);
                ВСЕ_ЕСЛИ
              ВСЕ_ЦИКЛ
            ВСЕ_ЦИКЛ
             ИНАЧЕ Печать («Пути из a в b не существует.»)
ВСЕ_ЕСЛИ
КОНЕЦ
```

Рис. 4.28. **Алгоритм нахождения минимального пути на неорграфе. (Волновой алгоритм)**

Алгоритм Форда для нагруженного орграфа.

Рассмотрим работу алгоритма на примере построения кратчайшего пути из вершины x_1 в вершину x_2 на графе G (рис. 4.29).

Разметим все вершины графа следующим образом: начальной вершине пути поставим метку $\boxed{0}$, всем остальным вершинам поставим метки $\boxed{\infty}$ (рис. 4.30-*а*). Далее будем просматривать все дуги графа, пока найдется хотя бы одна дуга (x,y), такая, что:

$$MV(y) > MV(x) + \text{Вес}(x,y) \qquad (4.2)$$

Если такая дуга (x,y) нашлась, то $MV(y) := MV(x) + \text{Вес}(x,y)$.

Рассмотрим дугу (x_1, x_2). $MV(x_2) > MV(x_1)$, по условию (4.2) $MV(x_2) := MV(x_1) + \text{Вес}(x_1, x_2)$ (рис. 4.30-*б*). Проверим все остальные дуги на выполнение условия (4.2) (рис. 4.30 *в*-*д*).

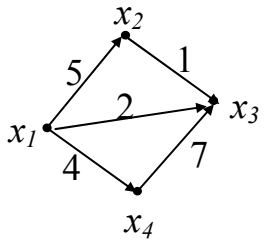


Рис. 4.29. **Нагруженный орграф *G***

На рис. 4.30-*д* показана окончательная разметка вершин графа G. Т.е. не осталось ни одной дуги, удовлетворяющей условию (4.2). Прямой ход алгоритма на этом закончен.

Обратный ход алгоритма строит путь между заданными вершинами. Будем считать конечную вершину пути текущей (4.30-*д*), обозначим ее за z. На рисунке текущая вершина выделена.

Следующую вершину минимального маршрута будем искать по условию:

Вершина x смежна с z и $\quad MV(x) = MV(z) - \text{Вес}(x, z) \qquad (4.3)$

Условию (4.3) удовлетворяет вершина x_3, перейдем к вершине x_3 (рис. 4.30-*е*).

Следующие вершины пути будем искать по условию (4.3), пока не дойдем до начальной вершины пути (рис. 4.30-*ж*). Минимальный путь из вершины x_1 в вершину x_2: $M = x_1\, x_3\, x_2$.

Алгоритм Форда, записанный на псевдокоде приведен на рис. 4.31.

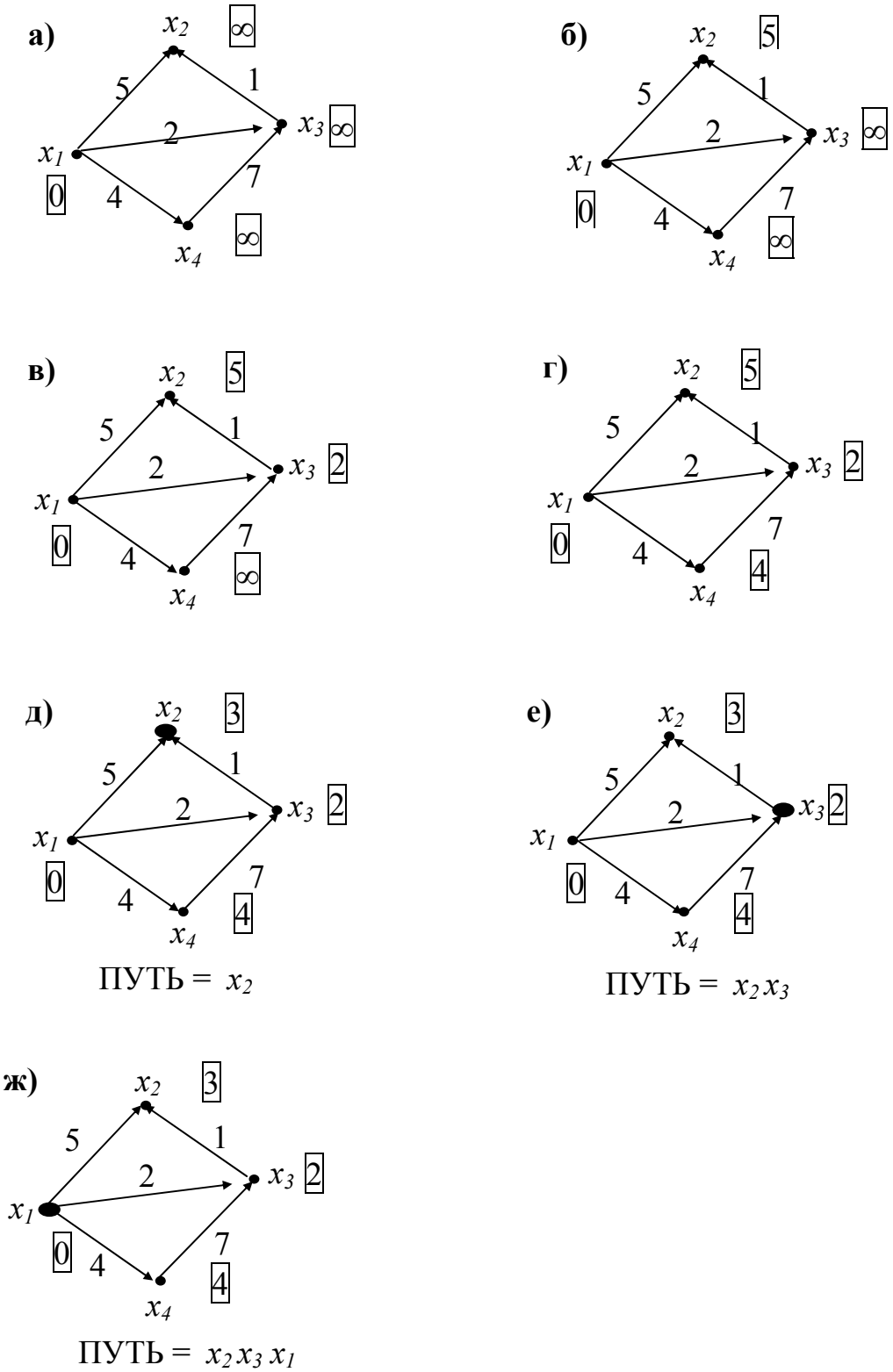


Рис. 4.30. **Поиск кратчайшего пути на нагруженном орграфе**

```
НАЧАЛО
( Разметка вершин)
MV(a):=0                          { Вершина a - начало пути }
 ЦИКЛ (для всех x из X\(a));   { Все остальные вершины пометить  }
    MV(x):= MaxInt;               { максимальным числом              }
 ВСЕ-ЦИКЛ
ЦИКЛ_ПОКА (найдется (x, y) из X, такая что  MV(x) - MV(y) > ВЕС(x, y))
       MV(y):=min (MV(y), MV(x)+ВЕС(x, y));
ВСЕ-ЦИКЛ
                  { По окончании разметки вершины размечены так, что }
                  { для всех (x, y)  MV(y) – MV(x) ≤ ВЕС(x, y) }
                        { Построение пути}
k:=MV(b);                 { Метка вершины b равна длине пути}
ПУТЬ(k):=b;
 ЦИКЛ_ПОКА (ПУТЬ(k) ≠ a)
     z:= ПУТЬ(k)
   ЕСЛИ СОСЕД (x, z) И
   MV(z) - MV(x) = ВЕС(x, z);
   ПУТЬ(k-1):= x;
   k:=k-1;
ВСЕ-ЦИКЛ
ВЫВОД  ПУТЬ(k),  k=0, 1,…, MV(b);
КОНЕЦ
```

Рис. 4.31. **Кратчайший путь на нагруженном орграфе из вершины *a* в вершину *b (алгоритм Форда)***

Алгоритм нахождения кратчайшего пути на любом нагруженном графе носит имя Дейкстры.

Алгоритм Дейкстры.

Рассмотрим алгоритм на примере графа G (рис. 4.32). Найдем кратчайший маршрут между вершинами x_1 и x_4. Выполним разметку вершин следующим образом. Начальной вершине x_1 поставим метку $\boxed{0}$, эту метку будем считать постоянной. Вершину x_1 будем считать текущей, $z:=x_1$. Всем остальным вершинам графа присвоим метки $\boxed{\infty}$, эти метки будем считать временными (рис. 4.33-*а*). На рис. 4.33 постоянные метки и текущая вершина выделены. Найдем все вершины x, смежные текущей вершине. Изменим метки этих вершин,

если $MV(x) > MV(z) + \text{Вес}(x, z)$, то

$$MV(x) = MV(z) + \text{Вес}(x, z). \qquad (4.4)$$

Для вершины x_1 смежные вершины x_2, x_6, x_3. Изменим метки вершин, как показано на рис. 4.33–*б*. Среди всех временных меток найдем минимальную. Это метка у вершины x_2. Сделаем вершину x_2 текущей, $z{:=}x_2$, метку вершины x_2 будем считать постоянной (рис. 4.33–*б*).

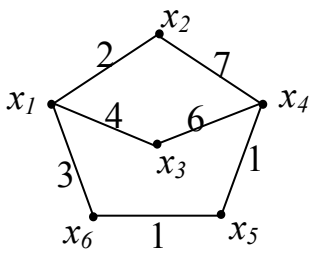


Рис. 4.32. **Нагруженный неорграф *G***

Пока текущей вершиной не станет конечная вершина маршрута, будем выполнять следующие действия:

- разметить по условию (4.4) все вершины, смежные текущей вершине z .
- из всех временных меток выбрать минимальную, соответствующую ей вершину сделать текущей, метку считать постоянной.

После того, как текущей вершиной станет конечная вершина, кратчайший путь построен. Его длина равна метке конечной вершине. Построение пути ведется аналогично обратному обходу алгоритма Форда.

Выполним эти действия для нашего примера. Для вершины x_2 найдем смежные. Это вершина x_4. После проверки условия (4.4) метка вершины x_4 станет равна 9 (рис. 4.33-*в*) Временные метки на этом шаге алгоритма имеют вершины x_6, x_3, x_4, x_5. Выберем минимальную временную метку. Это метка вершины x_6. Вершина x_6 становится текущей. Смежные вершины для текущей – x_1, x_5. Метка вершины x_1 не изменяется, метка вершины x_5 становится равной 4 (рис. 4.33-*г*) Вершины x_5, x_3, x_4 имеют временные метки. Минимальную метку имеют вершины x_3 и x_5. Выберем любую из этих вершин. Пусть это будет вершина x_3. Метка вершины x_3 становится постоянной. Смежные вершины для вершины x_3 – x_1 и x_4. Их метки не изменяются. (рис. 4.33-*д*). Среди вершин с временными метками минимальную метку имеет вершина x_5. Вершина x_5 становится текущей. Смежные ей вершины – x_6 и x_4. Метка вершины x_6 не изменяется. Метка вершины x_4 принимает значение 5 (рис. 4.33-*е*). В графе осталась одна вершина с временной меткой – вершина x_4. Поэтому вершиной с минимальной временной меткой является она же. Вершина x_4 становится текущей, ее метка становится постоянной (рис. 4.33-*ж*). На этом разметка вершин закончена.

Длина минимального пути из вершины x_1 в вершину x_4 равна значению метки вершины x_4, т.е. равна пяти. Построение пути согласно алгоритму Форда показано на рис. 4.33–*з*. Построенный путь: ПУТЬ=($x_1 x_6 x_5 x_4$).

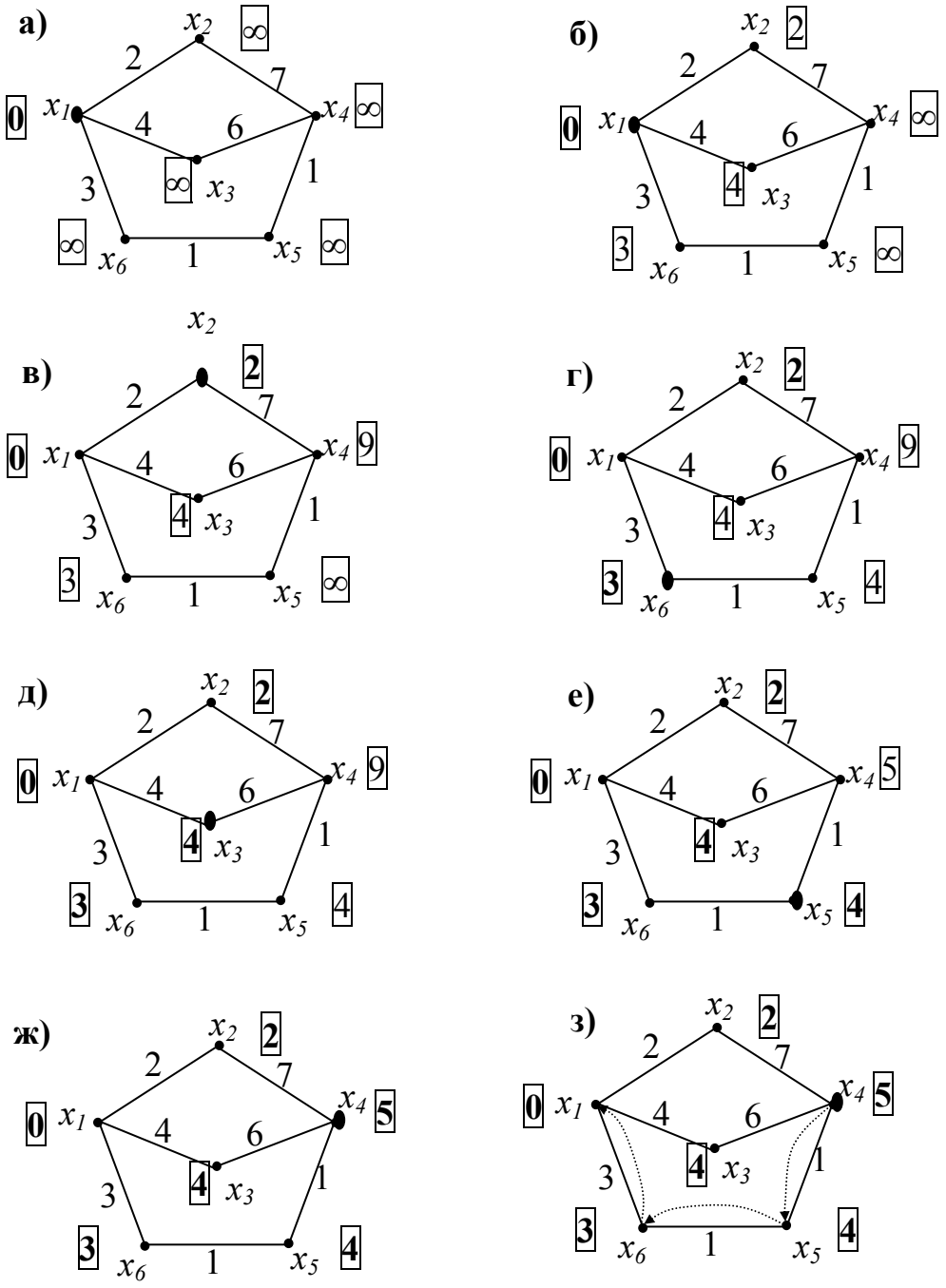


Рис. 4.33. **Алгоритм Дейкстры. Нахождение кратчайшего пути между вершинами x_1 и x_4**

Запишем прямой обход алгоритма Дейкстры на псевдокоде.

Обозначим за *a* начальную вершину маршрута, за *b* – конечную вершину маршрута.

Начнем разметку вершин. Будем записывать временные метки в массив В_Метка, постоянные метки в массив П_Метка. Метка вершины *a* постоянная и равна нулю. Метки всех остальных вершин временные и равны *MaxInt*.

П_Метка(a) := 0;
ЦИКЛ (для всех $x \in X \setminus a$) *{ для всех вершин графа, кроме начальной }*
  В_Метка(x) := MaxInt;
ВСЕ_ЦИКЛ

Обозначим текущую вершину за z. Текущая вершина – a.

z:=a;

Организуем цикл, пока текущая вершина не равна конечной вершине маршрута.

ЦИКЛ_ПОКА ($z \neq b$)
…
ВСЕ_ЦИКЛ

Для всех вершин графа, смежных с текущей вершиной, проверим условие (4.4).

ЦИКЛ (для всех x смежных с z)
  ЕСЛИ В_Метка(x) > В_Метка(z) + Вес(x, y)
        ТО В_Метка(x):= В_Метка(z) + Вес(x, y)
  ВСЕ_ЕСЛИ
ВСЕ_ЦИКЛ

Среди вершин с временными метками найдем вершину с минимальной меткой. Сделаем эту вершину текущей. Ее метку сделаем постоянной.

В_Метка[xx]:= min (В_Метка);
z:=xx;
П_Метка[xx]:= В_Метка[xx];

На рис. 4.34 прямой обход алгоритма Дейкстры представлен полностью.

4.7. Эйлеровы маршруты

Все не раз встречались с головоломками, решениями которых должна быть фигура, нарисованная без отрыва карандаша от бумаги, каждая линия которой не должна обводиться. Представим такую фигуру в виде графа.

Например, можно представить графом широко известную головоломку «Сабли Магомета» (рис. 4.35).

В терминах теории графов задача формулируется следующим образом: обойти все ребра графа ровно по одному разу.

Цикл, проходящий по всем ребрам графа ровно по одному разу, называется ***эйлеровым циклом***.

Цепь, проходящая по всем ребрам графа ровно по одному разу, называется ***эйлеровой цепью***.

```
П_Метка(a) := 0;
ЦИКЛ (для всех x ∈ X \ a)        { для всех вершин графа, кроме начальной }
  В_Метка(x) := MaxInt;
ВСЕ_ЦИКЛ
z:=a;
ЦИКЛ_ПОКА (z ≠ b)
  ЦИКЛ (для всех x смежных с z)
    ЕСЛИ  В_Метка(x) > В_Метка(z) + Вес(x, z)
          ТО  В_Метка(x):= В_Метка(z) + Вес(x, z)
    ВСЕ_ЕСЛИ
  ВСЕ_ЦИКЛ
  В_Метка[xx]:= min (В_Метка);
  z:=xx;
  П_Метка[xx]:= В_Метка[xx];
ВСЕ_ЦИКЛ
```

Рис. 4.34. **Прямой ход алгоритма Дейкстры.**

Теорема 4.3. Для того, чтобы в связном графе существовал эйлеров цикл необходимо и достаточно, чтобы степени всех вершин графа были четными.

Теорема 4.4. Для того, чтобы в связном графе существовала эйлерова цепь необходимо и достаточно, чтобы в графе было ровно две вершины нечетной степени.

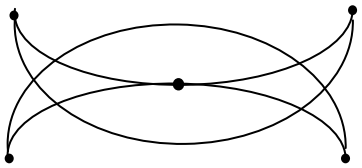


Рис. 4.35. **Сабли Магомета**

Рассмотрим *алгоритм построения эйлерова цикла* на графе, иллюстрирующем головоломку «Сабли Магомета» (рис. 4.35).

По теореме 4.3 в графе существует эйлеров цикл, если степени всех вершин четные. Степени вершин в выбранном графе четные: $\rho(1)=\rho(2)=\rho(4)=\rho(5)=2$; $\rho(3)=4$. Следовательно, возможно построить эйлеров цикл.

Будем строить в графе произвольным образом простой цикл. Ребра, вошедшие в построенный цикл, отметим метками 1 (рис. 4.36). На оставшихся

ребрах будем строить следующий цикл таким образом, чтобы хотя бы одна вершина второго и первого цикла была общей.

Ребра, вошедшие во второй цикл, отметим метками 2 (рис. 4.36).

Продолжаем построение циклов по этой схеме, пока не будут отмечены все ребра.

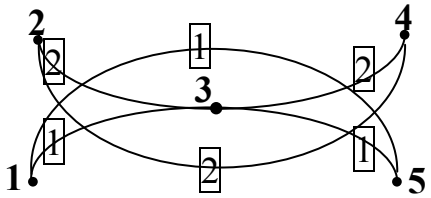


Рис. 4.36. **Разметка ребер для построения эйлерова цикла**

Для построения цикла выберем произвольную вершину графа. Будем обходить граф по следующему правилу: выход из текущей вершины всегда производится по ребру с максимальной меткой. Таким образом, получили цикл Э1=(1,3,4,2,3,5,1).

4.8. Решение задач 1-4 контрольной работы №4

Задача 1. Построить матрицы достижимости и взаимодостижимости графа *G* (рис. 4.37). Выделить сильные компоненты графа *G*, используя матрицу взаимодостижимости.

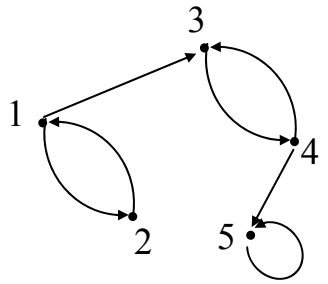


Рис. 4.37 **Задача 1. Орграф *G***

Решение. Пронумеруем произвольно вершины графа.

Построение матрицы достижимости *R*.

Размер матрицы (5×5). Из вершины 1 можно попасть в вершины 2, 3, 4, 5. По определению достижимости вершина 1 достижима сама из себя. Элементы первой строки матрицы достижимости равны единице.

Далее, из вершины 2 графа можно попасть в вершины 1, 2, 3, 4, 5. Вторая строка матрицы так же состоит из одних единиц.

Из вершины 3 можно попасть в вершины 3, 4 и 5. Элементы третьей строки матрицы с номерами 3, 4, 5 равны единице, остальные элементы третьей строки равны нулю.

Из вершины 4 можно попасть в вершины 3, 4, 5. Элементы с этими номерами в четвертой строке матрицы R равны единице. Из вершины 5 можно попасть только в вершину 5. Матрица достижимости R построена:

$$R = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Для того, чтобы построить матрицу взаимодостижимости, построим матрицу контрдостижимости P. Матрица контрдостижимости равна транспонированной матрице достижимости $P=R^T$.

$$P = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Элементы матрицы взаимодостижимости вычисляются по формуле: $s_{i,j}=r_{i,j}$ & $p_{i,j}$.

Матрица взаимодостижимости S:

$$S = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Выделим сильные компоненты графа по матрице взаимодостижимости.

Вычеркнем те столбцы и строки из матрицы S, которые соответствуют единичным элементам первой строки матрицы. Это элементы с номерами j=1 и j=2. Вычеркнем столбцы 1, 2 и строки 1,2. Таким образом, сильная компонента связности - подграф G_1 на вершинах 1,2 (рис. 4.38).

Далее рассмотрим измененную матрицу S.

$$S = \left[\begin{array}{c|ccc} & 3 & 4 & 5 \\ \hline 3 & 1 & 1 & 0 \\ 4 & 1 & 1 & 0 \\ 5 & 0 & 0 & 1 \end{array}\right].$$

Вычеркнем те столбцы и строки из матрицы *S*, которые соответствуют единичным элементам первой строки матрицы.

Это элементы с номерами *j*=3 и *j*=4. Вычеркнем столбцы 3, 4 и строки 3,4. Следующая сильная компонента связности подграф G_2 на вершинах 3,4 (рис. 4.38). После этого в измененной матрице *S* остался один элемент, это третья сильная компонента графа *G* – граф G_3 с одной вершиной 5.

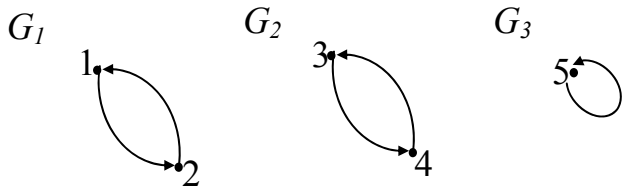


Рис. 4.38. **Сильные компоненты графа *G***

Задача 2. Найти диаметр, радиус, центры графа *G* (рис. 4.39).

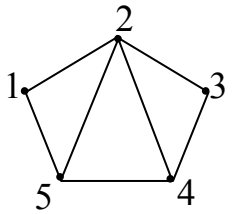


Рис. 4.39. **Задача 2. Граф *G***

Построим матрицу минимальных расстояний *D* графа *G*.

Длина минимального маршрута из вершины 1 в вершину 2 равна 1 (элемент матрицы $D - d_{1,2}$). Длина минимального маршрута из вершины 1 в вершину 3 равна 2 (элемент матрицы $D - d_{1,3}$).

Аналогично строятся все остальные элементы матрицы *D*.

Элементы главной диагонали матрицы равны нулю, т.к. длина пути из любой вершины в саму себя равна 0.

$$D = \begin{bmatrix} 0 & 1 & 2 & 2 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 0 & 1 \\ 1 & 1 & 2 & 1 & 0 \end{bmatrix} \quad F = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 2 \\ 2 \end{bmatrix}.$$

Выберем максимальные элементы из каждой строки матрицы D и запишем их в вектор F.

Максимальный элемент вектора F – диаметр графа G, равен 2, $D(G)=2$. Минимальный элемент вектора F – радиус графа G, равен 1, $r(G)=1$.

Если $f_i=r(G)$, то вершина i - центр графа. Тогда центром графа G является вершина 2.

Задача 3. Можно ли нарисовать граф G (рис. 4.40), не отрывая руки от бумаги, и не обводя дважды ни одно ребро. Обосновать ответ. Постройте пример эйлерова цикла или цепи, если это возможно.

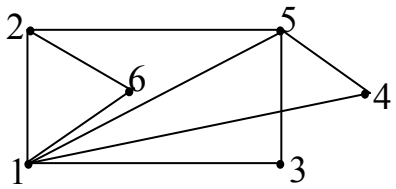


Рис. 4.40. **Задача 3. Граф *G***

Граф можно нарисовать, не отрывая руки от бумаги, и не обводя дважды ни одно из ребер, если выполняется одно из двух условий. По теореме Эйлера, в графе можно выделить эйлеров цикл, если степени всех вершин графа четные. Либо, если в графе ровно 2 вершины с нечетной степенью, то в графе можно построить эйлерову цепь.

Рассчитаем степени всех вершин графа. $\rho(1)=5$; $\rho(2)=3$; $\rho(3)=2$; $\rho(4)=2$; $\rho(5)=4$; $\rho(6)=2$. По теореме Эйлера в графе можно выделить эйлерову цепь, т.к. в графе ровно две вершины с нечетной степенью, это вершины 1 и 2. Построим эйлерову цепь. Начнем обводить граф с любой из вершин с нечетной степенью. Пусть это будет вершина 1. Тогда эйлерова цепь может быть записана следующим маршрутом E=(1,6,2,1,5,4,1,3,5,2).

Задача 4. Задача 4 контрольной работы подразумевает демонстрацию одного из алгоритмов, описанных в методическом пособии. Решением задачи считается изложение алгоритма и графическое изображение задачи, как показано в примерах к каждому алгоритму. Т.е. подробное изложение алгоритма для конкретного графа, с указанием значений всех переменных, использующихся в алгоритме.

4.9. Контрольные вопросы и упражнения

1. Постройте несколько цепей из вершины 1 в вершину 4 на графе *G* (рис. 4.40). Запишите маршруты различными способами. Определите, какие из построенных цепей простые, а какие нет.

2. Постройте несколько контуров на графе *G* (рис. 4.41). Запишите маршруты различными способами. Определите, какие из построенных контуров простые, а какие нет.

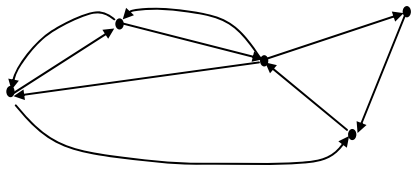
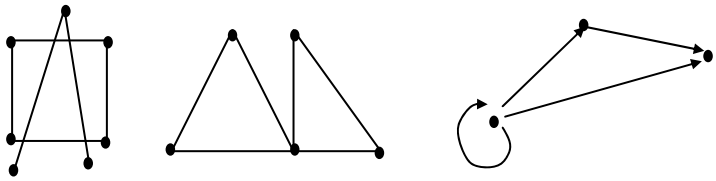
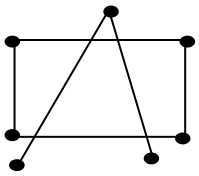


Рис. 4.41. **Граф *G***

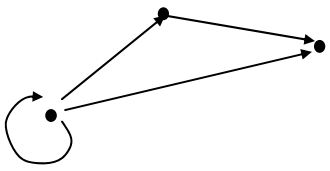
3. Постройте множества достижимости, контрдостижимости для графов:



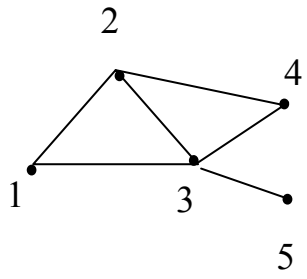
4. Сколько компонент связности имеет граф:



5. Определить связность графа (сильно связный, слабо связный, не связный). Выделить компоненты сильной связности.



6. Продемонстрируйте алгоритм обхода графа «в глубину», если стоит задача найти вершину 4, начиная обход из вершины 1.



7. Укажите отличия между алгоритмами Форда и Дейкстры.

8. Выберите правильные по вашему мнению утверждения:

а) волновой алгоритм строит кратчайший путь между двумя вершинами на нагруженном орграфе;

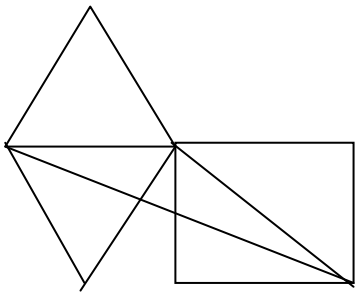
б) обход графа в глубину можно использовать для выделения компонент связности;

в) алгоритм Форда строит кратчайший путь между двумя вершинами на связном неорграфе;

г) алгоритм Дейкстры может использоваться для любых типов графов, как для орграфов, так и для неорграфов.

9. Продемонстрируйте работу алгоритма Дейкстры для нахождения кратчайшего пути в графе на рис. 4.29 из вершины x_1 в вершину x_2.

10. Постройте пример эйлерова цикла, используя описанный алгоритм, в графе:



11. Реализуйте рассмотренные алгоритмы на языке Паскаль, изменяя способы задания графа.

ГЛАВА 5. ДЕРЕВЬЯ И ЦИКЛЫ

5.1. Основные определения

В отдельный класс выделены графы без циклов.
Связный граф без циклов называется ***деревом*** (рис. 5.1).

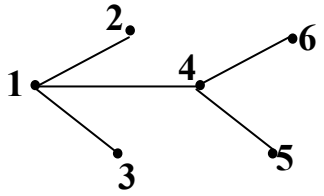


Рис. 5.1. **Граф – дерево**

Несвязный граф без циклов называется ***лесом*** (рис. 5.2).

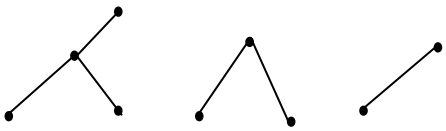
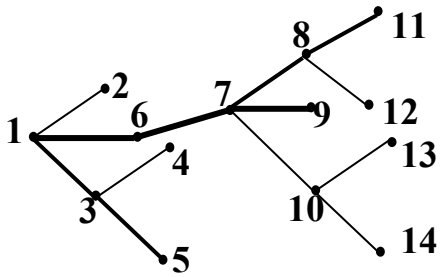


Рис. 5.2. **Граф – лес**

Висячие вершины в деревьях называются ***листьями***. В дереве на рис. 5.1 вершины 2, 3, 6, 5 – листья. Если в дереве выделена вершина k, как вершина 1 (рис. 5.1), то такую вершину назовем ***корнем*** дерева. В деревьях используют такое понятие как ***ствол*** (рис. 5.3). Любая цепь между вершиной, принадлежащей стволу дерева, и листом дерева называется ***ветвью***.



Цепь (1,6,7,9) – ствол дерева,
цепи (1,3,5) и (7,8,11) – ветви.

Рис. 5.3. **Стволы и ветви в деревьях**

5.2. Свойства деревьев

Теорема 5.1. Любое дерево на n вершинах содержит n-1 ребер.

Доказательство. Докажем теорему методом математической индукции (ММИ).

Выберем основание индукции n=2. Очевидно, что в дереве с n=2 вершинами одно ребро.

Предположим, что в дереве с n-1 вершинами n-2 ребер.

Добавим одну вершину в дерево. Число вершин n. Т.к. дерево связный граф, то необходимо добавить хотя бы одно ребро, связывающее добавленную вершину с любой другой вершиной дерева. Если добавленную вершину связать с несколькими, т.е. добавить более чем одно ребро, то в графе появится цикл. Следовательно, в дереве с n вершинами может быть только n-1 ребер.

Следствие теоремы 5.1. Всякий лес содержит ровно n-k ребер, где k – число компонент связности.

Теорема 5.2. Любая цепь в дереве будет простой цепью.

Доказательство. Пусть цепь $x_1 \ldots x_k$, выделенная в дереве, не простая цепь. Тогда, по определению, найдутся хотя бы две вершины $x_i = x_j$, $i \neq j$. В этом случае в графе можно выделить цикл, что противоречит первоначальному условию, что граф – дерево.

Теорема 5.3. Любые две вершины дерева соединены простой и единственной цепью.

Доказательство. Т.к. по определению дерево – связный граф, то любые две его вершины соединены. По *теореме 5.2*, любая цепь, выделенная в дереве, будет простой цепью. Предположим, что две вершины дерева x и y соединены двумя цепями. Но в этом случае в графе можно выделить цикл, а это противоречит условию, что граф – дерево.

Теорема 5.4. При добавлении ребра между любыми двумя вершинами графа-дерева, в графе появляется ровно один цикл.

Доказательство. По *теореме 5.1* в дереве с n вершинами ровно n-1 ребер. Добавление ребра, таким образом, обязательно приведет к образованию цикла. Докажем теперь, что этот цикл будет единственным.

При n=3 очевидно, что добавление ребра между вершинами x и y приведет к появлению единственного цикла (рис. 5.4).

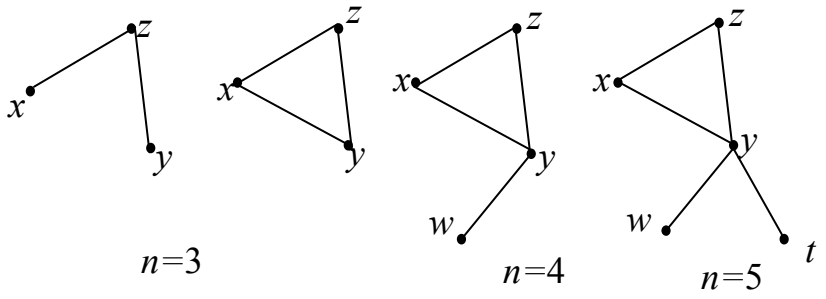


Рис. 5.4. **Иллюстрация к *теореме 5.4***

Построим граф с n=4 вершинами, таким образом: добавим одну вершину и соединим ее с любой из трех вершин одним ребром. Тогда удалив ребро между вершинами x и y, получим граф без циклов, т.е. дерево (рис. 5.4). Строя, таким образом, графы на 4, 5, … n вершинах получим графы с единственным циклом. Причем удаление ребра между вершинами x и y приводит к тому, что граф становится деревом.

Теорема 5.5. При удалении любого ребра из дерева число компонент связности увеличивается на единицу.

Доказательство. По *теореме 5.3* любые две вершины дерева соединены единственной простой цепью, таким образом, удаление любого ребра приведет к тому, что граф станет не связным. Докажите самостоятельно, используя способ *теоремы 5.4*, что число компонент связности изменится ровно на единицу.

Теорема 5.6. В дереве существуют хотя бы две висячие вершины.

Доказательство. Очевидно, что при n=2, обе вершины висячие. Построим дерево с n=3 вершинами. В таком дереве так же две вершины висячие. Добавим в дерево еще одну вершину, n=4. Для того, чтобы граф остался связным, мы должны добавить еще одно ребро, связывающее новую добавленную вершину с произвольной вершиной графа. Т.к. граф с n=4 должен остаться деревом, то это ребро будет единственным. Тогда висячей обязательно станет добавленная вершина. Если добавленное ребро соединяет новую вершину с висячей вершиной, то число висячих вершин не изменится, если же нет, то число висячих вершин увеличится. А значит, в дереве останется хотя бы две висячие вершины. Строя, таким образом, графы с 5, 6, …, n вершинами, получим деревья с числом висячих вершин не меньше, чем две.

5.3. Кодирование деревьев

Теорема 5.7. Число деревьев построенных на n вершинах равно:

$$L = n^{n-2}.$$

Доказательство. Будем считать, что нумерация вершин дерева остается неизменной при различном соединении вершин. Действительно, при n=2 можно построить лишь одно дерево (рис. 5.5-*а*), при n=3 можно построить три дерева (рис. 5.5-*б*).

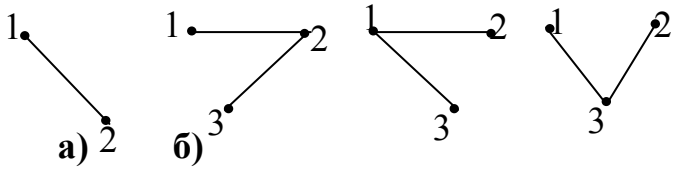


Рис. 5.5. **Построение деревьев:**
а) дерево на n=2 вершинах; б) деревья на n=3 вершинах

Рассмотрим дерево G с n вершинами. Пронумеруем вершины 1, 2, …, n. По *теореме 5.6* в дереве обязательно существуют висячие вершины. Выберем из всех висячих вершин вершину с минимальным номером. Обозначим ее за b_1. Рассмотрим ребро, инцидентное b_1 - $v_1=(b_1, a_1)$. Вторую концевую вершину этого ребра обозначим за a_1. Запишем вершину a_1 в вектор $T=[a_1]$. Удалим из дерева вершину b_1 и ребро v_1.

Полученное дерево назовем так же – G. В дереве G найдем следующую висячую вершину с минимальным номером. Обозначим ее за b_2. Рассмотрим ребро, инцидентное $b_2 – v_2=(b_2, a_2)$. Вторую концевую вершину этого ребра обозначим за a_1. Запишем вершину a_2 в вектор $T=[a_1, a_2]$. Удалим из дерева вершину b_2 и ребро v_2.

Далее будем рассматривать измененное дерево. Описанные действия будем продолжать до тех пор, пока в дереве не останется одно ребро. Вектор $T=[a_1, a_2,…,a_{n-2}]$ состоит из n-2 элементов.

Дерево G на n вершинах однозначно определяется вектором T. Вектор T называется ***кодом дерева***.

Рассмотрим все возможные варианты записи вектора T. Это размещения с повторениями из n элементов по n-2 элементов. Тогда число всех возможных вариантов вектора T равно $\overline{A}_n^{n-2} = n^{n-2}$. Следовательно, число деревьев построенных на n вершинах равно n^{n-2}.

Теорема 5.7 описывает способ представления деревьев кодом T. Такую процедуру называют ***кодированием дерева***. Построим код дерева G (рис. 5.6).

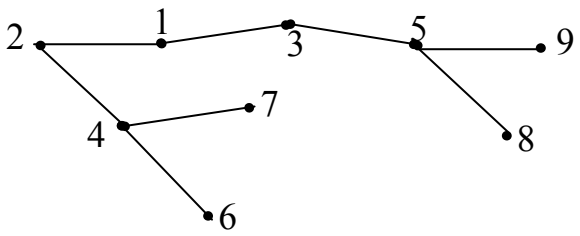
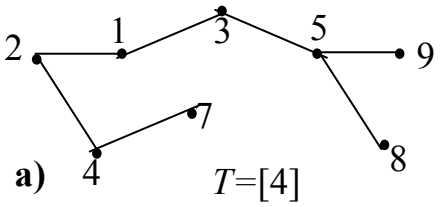


Рис. 5.6. **Дерево *G***

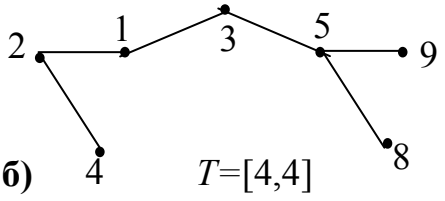
Найдем висячую вершину с минимальным номером. Это вершина 6. Ей инцидентно ребро (6,4). Запишем вершину 4 в код дерева T – T=[4]. Удалим из дерева вершину 6 и ребро (6,4) (рис. 5.7-*а*). Следующая висячая вершина с минимальным номером – вершина 7. Ей инцидентно ребро (7,4). Вершину 4 запишем в код дерева T=[4,4]. Удалим из дерева вершину 7 и ребро (7,4). На рис. 5.7 показано построение кода дерева.

Можно выполнить обратную процедуру – построение дерева по его коду. Рассмотрим построение дерева по коду T=[3,3,7,1,2,6]. В коде дерева 6 элементов, следовательно, в дереве 8 вершин. Изобразим эти 8 вершин и пронумеруем их в произвольном порядке (рис. 5.8-*а*).

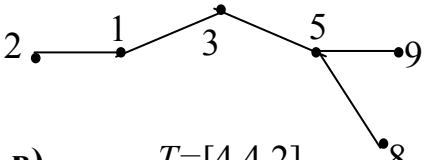
Найдем вершины, не вошедшие в код дерева. Это вершины 4, 5, 8. Из условия построения кода это висячие вершины. Создадим множество W={4,5,8}, содержащее эти вершины. Из вершин множества W выберем вершину с минимальным номером. Это вершина 4. Соединим вершину 4 с первым элементом кода дерева, т.е. с вершиной 3.



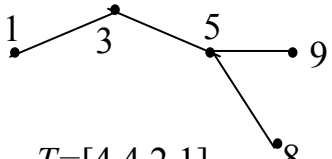
а) T=[4]



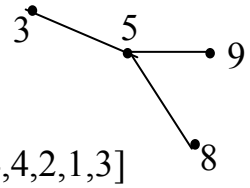
б) T=[4,4]



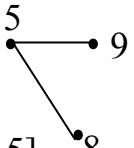
в) T=[4,4,2]



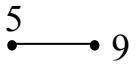
г) T=[4,4,2,1]



д) T=[4,4,2,1,3]



е) T=[4,4,2,1,3,5]



ж) T=[4,4,2,1,3,5,5]

Рис. 5.7. **Построение кода дерева**

Соединение вершин 4 и 3 показано на рис. 5.8-*б*. Удалим вершину 4 из W и вершину 3 из T. W={5,8}, T=[3,7,1,2,6]. Выберем следующую вершину с минимальным номером из W. Это вершина 5. Соединим вершину 5 с первым элементом вектора T – вершиной 3 (рис. 5.8-*в*). Удалим вершину 3 из кода дерева, вершину 5 из множества W. Вершина 3 больше не встречается в коде дерева ни разу, поэтому запишем вершину 3 во множество W. W={3,8}, T=[7,1,2,6]. Следующая вершина с минимальным номером из W - вершина 3. Вершину 3 следует соединить с вершиной 7 – первым элементом кода дерева (рис. 5.8-*г*). После этого вершина 3 удаляется из W, а вершина 7 удаляется из кода дерева и записывается в W. W={7,8}, T=[1,2,6].

Таким образом при добавлении ребра в дерево выполняются следующие действия:

- i - вершина с минимальным номером в W;
- j - первый по порядку элемент кода дерева T;
- соединить вершины i и j (дописать ребро (i, j) в дерево);
- удалить вершину i из W и вершину j из T, если j более не встречается в коде дерева, то дописать j в W;
- после того, как код дерева стал пустым, во множестве W останется два элемента $W=\{k, l\}$, добавим в дерево ребро (k, l). Удалим эти элементы из множества W;
- построение дерева закончено.

Дальнейшее построение дерева показано на рис. 5.8 *д-з.* На рис. 5.8-*и* построенное дерево представлено в плоском виде.

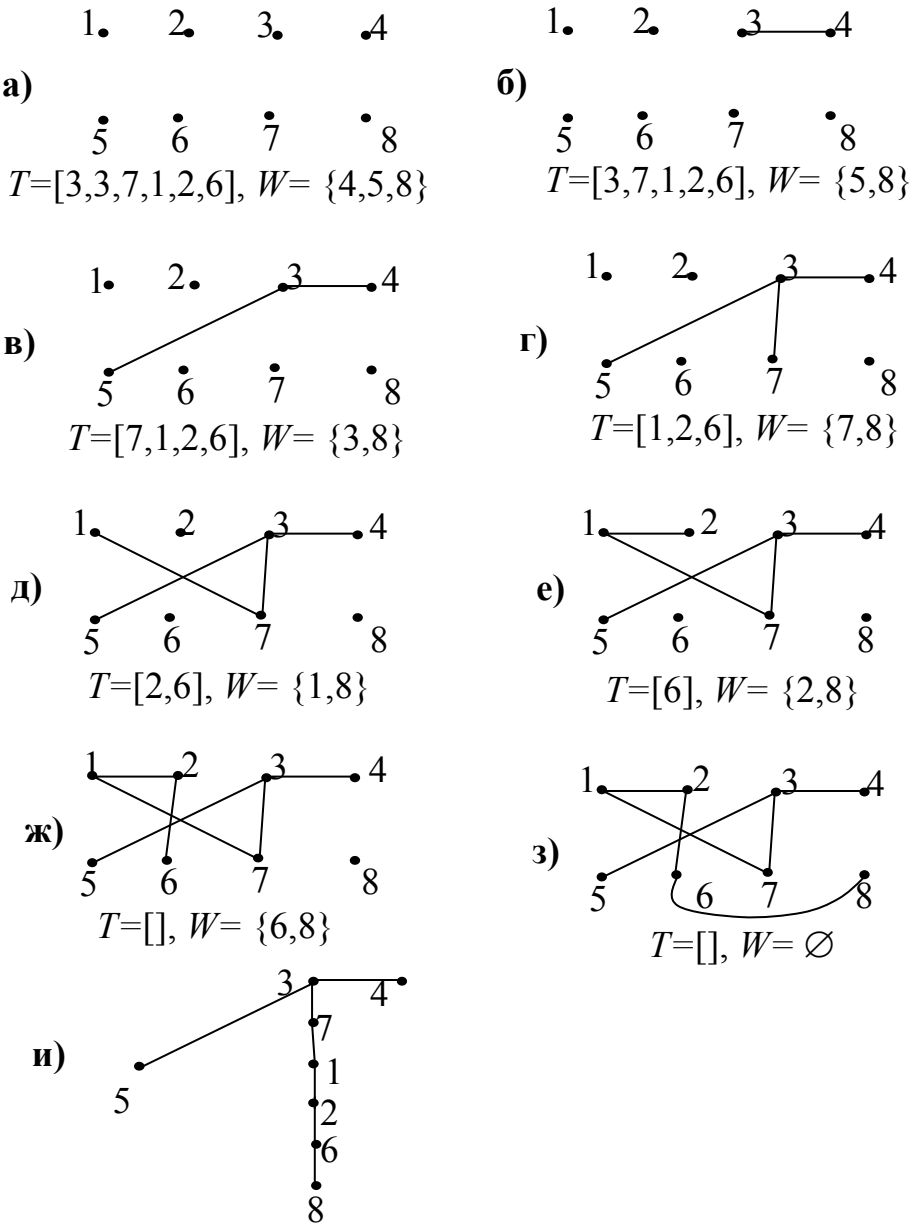


Рис. 5.8. **Построение дерева по его коду**

5.4. Обходы дерева

Определим, что обойти дерево, значит, составить маршрут, который обходил бы все его вершины. Во время обхода каждой вершине будем каждой вершине дерева ставить метку - 0, 1, 2, …, n. Где n – количество вершин дерева. Существует два способа обхода деревьев – обход «в глубину» и обход «в ширину».

Обход дерева «в глубину».

С понятием обхода графа «в глубину» мы уже встречались ранее. Если рассматривать дерево с корнем, то за один прямой ход (без возвратов) будет построена ветвь дерева, т.е. цепь от корня до листа. Далее пройденная ветвь дерева рассматривается как ствол, и следующая ветвь строится от этого ствола и т.д.

Обойдем дерево G (рис. 5.9), используя обход «в глубину».

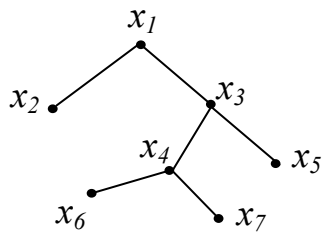


Рис. 5.9. **Дерево *G***

Выберем произвольную вершину в дереве. Пусть это будет вершина x_1. Поставим вершине x_1 метку $\boxed{0}$. Всем остальным вершинам поставим метки, равные $\boxed{-1}$. Вершина x_1 считается текущей вершиной. Запишем текущую вершину в стек (рис. 5.10-*а*). Выберем произвольно вершину дерева, такую, что она смежна текущей вершине x_1 и не была ранее пройдена, т.е. ее метка равна $\boxed{-1}$. Пусть это будет вершина x_3. Вершина x_3 становится текущей и записывается в стек. Ее метка становится равной $\boxed{1}$ (рис. 5.10-*б*). Следующая вершина, смежная текущей и метка которой равна $\boxed{-1}$ – вершина x_5. Вершина становится текущей и записывается в стек. Метка вершины x_5 – $\boxed{2}$ (рис. 5.10-*в*). У вершины x_5 нет смежных не пройденных вершин. Отметим, что существование не пройденного ребра для деревьев не проверяется, т.к. в деревьях нет циклов, то такие случаи исключаются.

Используя стек, начнем обратный обход. Удалим из стека вершину x_5 и перейдем к вершине, прочитанной из стека. Это вершина x_3 (рис. 5.10-*г*). У вершины x_3 существует смежная, не пройденная ранее вершина – x_4. Запишем вершину x_4 в стек. Метка вершины x_4 – $\boxed{3}$ (рис. 5.10-*д*). Обход дерева заканчивается, когда все вершины пройдены. Дальнейший обход дерева представлен на рис. 5.10 *е-м*. Запишите самостоятельно алгоритм обхода дерева на псевдокоде.

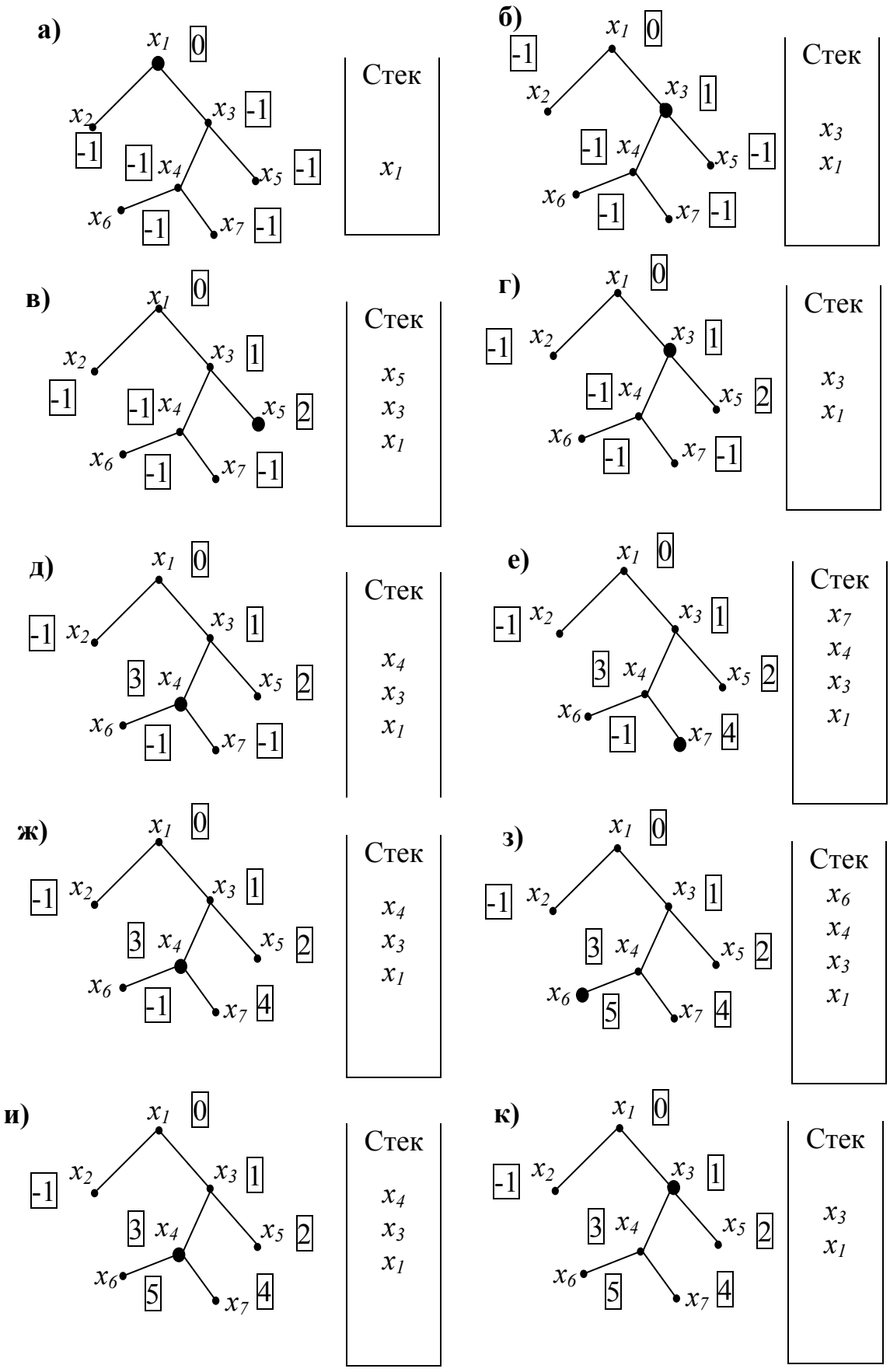
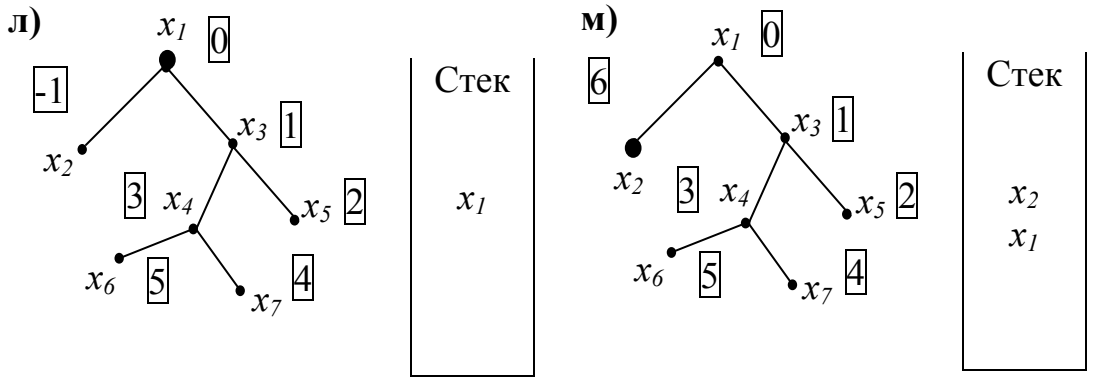


Рис. 5.10. **Обход дерева «в глубину»**



Продолжение рис. 5.10. **Обход дерева «в глубину»**

Обход дерева «в ширину».

Визуально дерево можно разбить на несколько «уровней» (рис. 5.11).

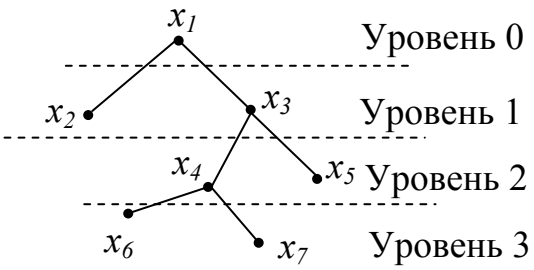


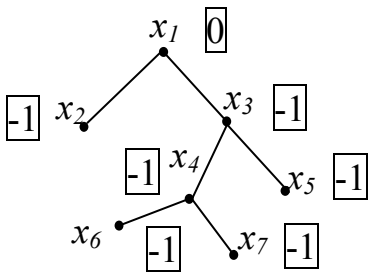
Рис. 5.11

Смысл обхода «в ширину» - отмечать все множество вершин одного «уровня». Рассмотренный ранее волновой алгоритм – модификация обхода графа в ширину.

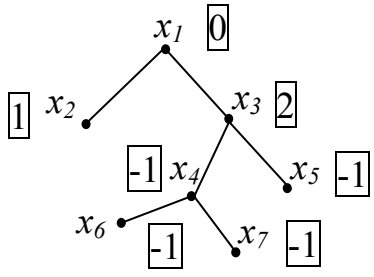
Рассмотрим обход в ширину на примере обхода дерева G (рис. 5.9).

Выберем произвольно начальную вершину обхода. Пусть это будет вершина x_1. Пусть k=0. Поставим текущей вершине x_1 метку, равную 0. Вершину x_1 занесем во множество $U_k=U_0=\{x_1\}$ (рис. 5.12-*а*).

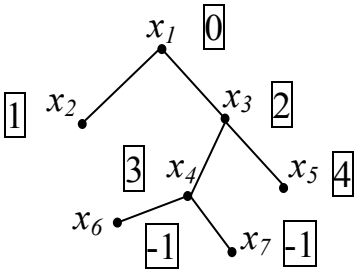
Найдем вершины, смежные вершинам из множества $U_k=U_0$. Это вершины x_2 и x_3. Запишем их во множество $U_{k+1}=U_1=\{x_2, x_3\}$. Поставим вершине x_2 метку 1, вершине x_3 - метку 2 (рис. 5.12-*б*). Увеличим значение k на единицу - k:=k+1. Найдем вершины, смежные вершинам из множества $U_k=U_1$. Это вершины x_4 и x_5. Запишем их во множество $U_{k+1}=U_2=\{x_4, x_5\}$. Поставим вершине x_4 метку 3, вершине x_5 - метку 4 (рис. 5.12-*в*). Увеличим значение k на единицу - k:=k+1. Найдем вершины, смежные вершинам из множества $U_k=U_2$. Это вершины x_6 и x_7. Запишем их во множество $U_{k+1}=U_3=\{x_6, x_7\}$. Поставим вершине x_6 метку 5, вершине x_7 - 6 (рис. 5.12-*ж*). Алгоритм заканчивает свою работу после того, как были отмечены все вершины дерева.



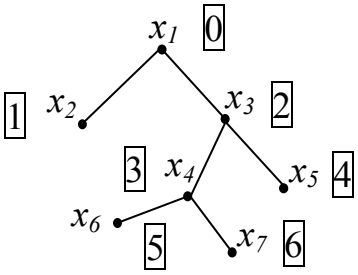
а) k=0, $U_k=\{x_1\}$



б) k=1, $U_0=\{x_1\}$, $U_1=\{x_2, x_3\}$



в) k=2, $U_0=\{x_1\}$, $U_1=\{x_2, x_3\}$
$U_2=\{x_4, x_5\}$



г) k=3, $U_0=\{x_1\}$, $U_1=\{x_2, x_3\}$
$U_2=\{x_4, x_5\}$, $U_3=\{x_6, x_7\}$

Рис. 5.12. **Обход графа «в ширину»**

Используя волновой алгоритм, запишите самостоятельно алгоритм обхода графа «в ширину».

5.5. Цикломатическое число

Одной из характеристик графа является цикломатическое число.
Цикломатическим числом графа $G(X, V)$ называется величина:

$$\lambda(G) = m - n + k(G), \qquad (5.1)$$

где $m = |V|$, $n = |X|$, $k(G)$ – число компонент связности G.

Найдем цикломатические числа для графов на рис. 5.13.

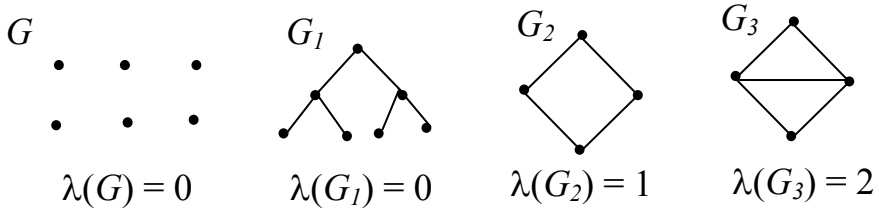


Рис. 5.13. **Графы и их цикломатические числа**

Для графа G (рис. 5.13) - $n = 6$, $m = 0$, $k(G) = 6$, $\lambda(G) = 0 - 6 + 6 = 0$.
Для графа G_1 (рис. 5.13) - $n = 7$, $m = 6$, $k(G) = 1$, $\lambda(G) = 6 - 7 + 1 = 0$.
Для графа G_2 (рис. 5.13) - $n = 4$, $m = 4$, $k(G) = 1$, $\lambda(G) = 4 - 4 + 1 = 1$.
Для графа G_3 (рис. 5.13) - $n = 4$, $m = 5$, $k(G) = 1$, $\lambda(G) = 5 - 4 + 1 = 1$.

Заметим, что если в графе нет циклов, то цикломатическое число равно нулю. Графы G и G_1 (рис. 5.13) не имеют циклов.

Если ребро графа входит хотя бы в один простой цикл, такое ребро называют ***цикловым***.

Перешейком называют ребро, не вошедшее ни в один из простых циклов.

В графе G (рис. 5.14) все ребра цикловые, в графе G_1 (рис.5.14) все ребра перешейки, в графе G_2 ребра I, II, III, IV, VI, VII, VIII, IX – цикловые, ребро V– перешеек.

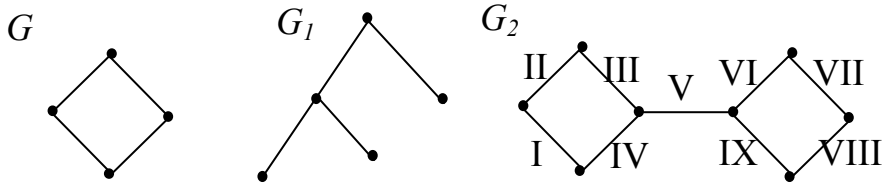


Рис. 5.14. **Неорграфы**

При удалении циклового ребра связность графа не изменяется. При удалении перешейка число компонент связности графа увеличивается на единицу (рис. 5.15).

Цикломатическое число графа при удалении перешейка не изменяется. При удалении циклового ребра цикломатическое число уменьшается на единицу (рис. 5.15).

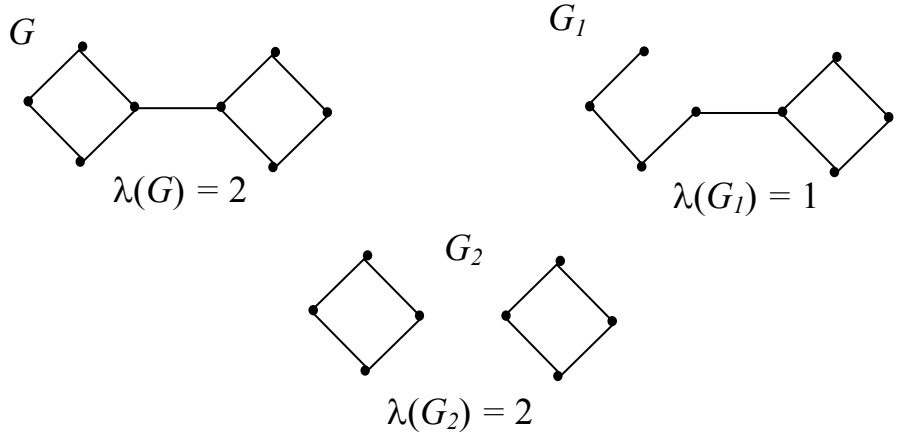


Рис. 5.15. **Удаление ребер и цикломатическое число**

$$\lambda(G) = 9 - 8 + 1 = 2, \quad \lambda(G_1) = 8 - 8 + 1 = 1, \quad \lambda(G_2) = 8 - 8 + 2 = 2.$$

Цикломатическое число графа всегда больше или равно нулю. Докажем это утверждение.

Пусть цикломатическое число графа > 0. Удалим из графа все цикловые ребра. Цикломатическое число такого графа равно нулю. Если после удаления в графе остались какие-либо ребра, то эти ребра являются перешейками. Удаление перешейка не изменяет цикломатического числа, т.к. при удалении перешейка уменьшается число ребер на единицу, $m' = m - 1$, и увеличивается число компонент связности, $k'(G) = k(G) + 1$.

Подставим эти значения в формулу 5.1:

$$\lambda(G') = (m - 1) - n + (k(G) + 1) = m - n + k(G) = 0.$$

Следовательно, цикломатическое число графа всегда положительное число.

Пусть $G(X,V)$ несвязный граф с k компонентами связности. Тогда цикломатическое число графа $G(X,V)$ равно сумме цикломатических чисел всех компонент связности графа:

$$\lambda(G) = \sum_{i=1}^{k} \lambda(G_i),$$

где G_i - i-тая компонента связности.

Для графа G (рис. 5.16) $\lambda(G) = 10 - 10 + 3 = 3$.

Цикломатические числа компонент связности графа G (рис. 5.16):

$$\lambda(G_1) = 5 - 4 + 1 = 2, \quad \lambda(G_2) = 1 - 2 + 1 = 0,$$
$$\lambda(G_3) = 4 - 4 + 1 = 1, \quad \lambda(G) = 2 + 0 + 1 = 3.$$

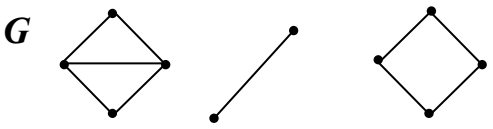


Рис. 5.16. **Несвязный граф**

Докажем это утверждение.

Число ребер графа G равно сумме ребер всех компонент связности:

$$m = m_1 + m_2 + ... + m_k = \sum_{i=1}^{k} m_i \qquad (5.2)$$

Число вершин графа G равно сумме вершин всех компонент связности:

$$n = n_1 + n_2 + ... + n_k = \sum_{i=1}^{k} n_i \qquad (5.3)$$

Каждая из компонент связности графа G есть связный подграф, т.е. $k(G_i) = 1$, $i = \overline{1,k}$. Тогда:

$$k = k(G_1) + k(G_2) + ... + k(G_k) = \sum_{i=1}^{k} 1 \qquad (5.4)$$

Найдем сумму цикломатических чисел всех компонент связности графа G:

$$(m_1 - n_1 + 1) + (m_2 - n_2 + 1) + ... + (m_k - n_k + 1) = \sum_{i=1}^{k} m_i - \sum_{i=1}^{k} n_i + k \qquad (5.5)$$

Заменим суммы выражения (5.5) на левые части выражений (5.2) и (5.3).

$$\sum_{i=1}^{k} \lambda(G_i) = m - n + k = \lambda(G)$$

5.6. Остовные деревья

Остовным деревом связного графа *G(X,V)* будем называть любой суграф графа *G(X,V)* без циклов (рис. 5.17).

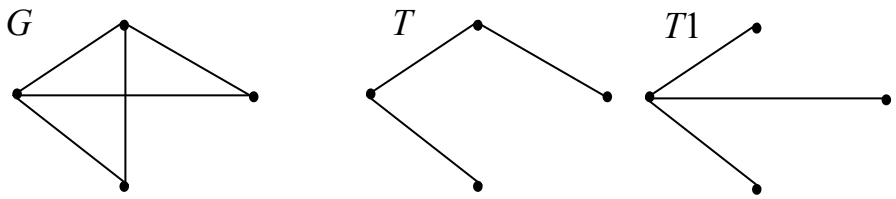


Рис. 5.17. **Граф *G* и его остовные деревья *T* и *T*1**

Чтобы получить остовное дерево графа, из графа *G(X,V)* нужно удалить ровно $\lambda(G)$ребер.

Докажем это утверждение. По определению в дереве на n вершинах n-1 ребро. Тогда из графа *G(X,V)* нужно удалить $m - (n + 1) = m - n + 1$.

Т.к. по определению граф *G(X,V)* связный, то это и есть цикломатическое число графа.

Очевидно, что граф может иметь несколько различных остовных деревьев (рис. 5.17).

Рассмотрим два способа построения остовных деревьев.

Построение остовного дерева графа с помощью алгоритма обхода «в глубину».

Построим остовное дерево графа *G* (рис. 5.18)

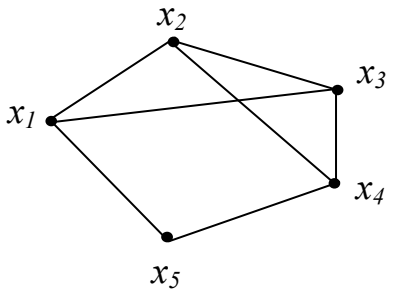


Рис. 5.18. **Неорграф *G***

Остовное дерево графа *G* (рис. 5.18) – граф-дерево $T(X,V')$, $|X|=5$, $|V'|=4$. Действительно, цикломатическое число графа *G* - $\lambda(G) = 7 - 5 + 1 = 3$, т.е. из графа *G* для получения остовного дерева необходимо удалить три ребра.

Начальная вершина обхода выбирается произвольно. Пусть это будет вершина x_1. Поставим вершине x_1 метку $\boxed{0}$, всем остальным вершинам графа метки, равные $\boxed{-1}$. Вершина x_1 считается текущей, записывается в стек. Будем записывать в массив *TREE* ребра, включаемые в остовное дерево *T*. В начале алгоритма $TREE = \varnothing$ (рис. 5.19-*а*).

Найдем вершину, которая ранее не проходилась и была бы смежной текущей. Т.е. метка вершины < 0. Таких вершин несколько. Это вершины x_2, x_3, x_5. Выберем одну из них произвольно. Пусть это будет вершина x_3. Ребро (x_1, x_3) записывается в массив *TREE*, вершина x_3 становится текущей. Сама вершина записывается в стек и ей ставится метка, равная $\boxed{1}$ (рис. 5.19-*б*).

Для текущей вершины x_3 опять найдем смежные, ранее не пройденные вершины. Это вершины x_2, x_4. Выберем из них, например, вершину x_4. Ребро (x_3, x_4) записывается в массив *TREE*. Сама вершина записывается в стек и ей ставится метка, равная $\boxed{2}$ (рис. 5.19-*в*). Для текущей вершины x_4 смежная и ранее не пройденная вершина x_5. Ребро (x_4, x_5) записывается в массив *TREE*. Сама вершина записывается в стек и ей ставится метка, равная $\boxed{3}$ (рис. 5.19-*г*).

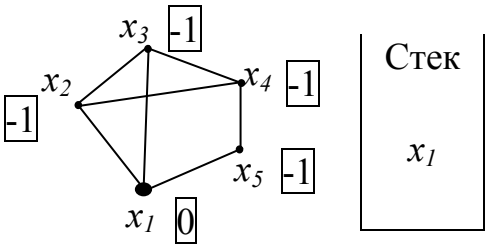
У текущей вершины x_5 нет смежных вершин с метками, равными $\boxed{-1}$, но еще не пройдены все вершины графа *G*. Поэтому, начнем обратный обход, используя информацию, хранящуюся в стеке. Удалим вершину x_5 из стека и перейдем к вершине x_4 (рис. 5.19-*д*). У текущей вершины x_4 есть смежная вершина с меткой $\boxed{-1}$. Это вершина x_2. Запишем вершину x_2 в стек, поставим ей метку $\boxed{4}$. Ребро (x_4, x_2) запишем в массив *TREE*. Вершину x_2 сделаем текущей (рис. 5.19-*е*).

На этом построение остова графа заканчивается, т.к. были пройдены все вершины графа. Остовное дерево графа G представлено на рис. 5.19-*ж*.

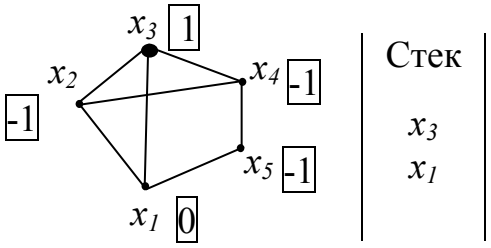
Запишем алгоритм построения остовного дерева, используя ранее написанный алгоритм обхода «в глубину».

Пусть a – начальная вершина обхода. Метка начальной вершины равна 0, метки всех остальных вершин равны –1. Стек пустой. В переменной m_v будем хранить значение последней поставленной метки, отличной от -1. Множество ребер остовного дерева *TREE* – пустое множество.

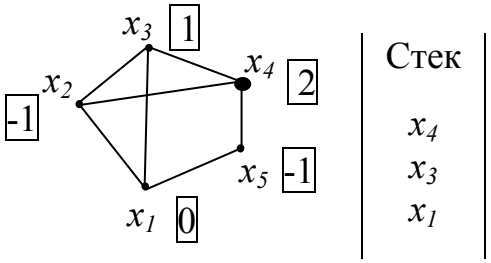
ЦИКЛ (для всех $x \in X$)



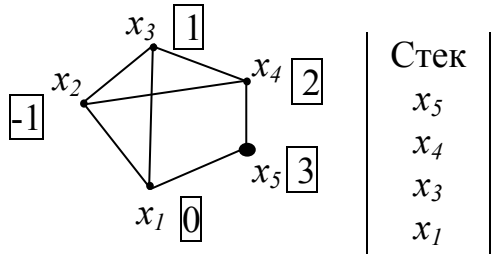
а) $TREE = \varnothing$



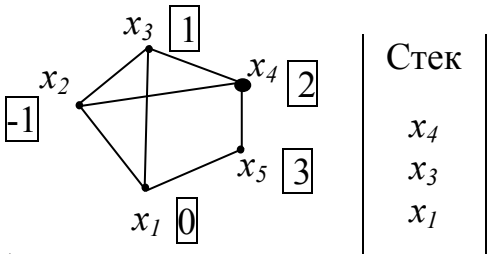
б) $TREE = \{(x_1, x_3)\}$



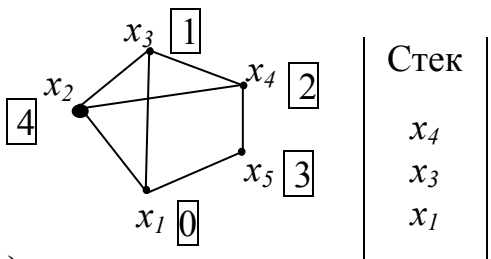
в) $TREE = \{(x_1, x_3), (x_3, x_4)\}$



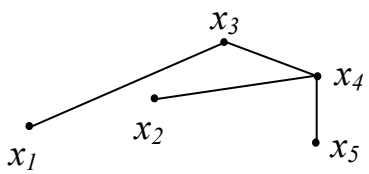
г) $TREE = \left\{ \begin{matrix} (x_1, x_3), (x_3, x_4), \\ (x_4, x_5) \end{matrix} \right\}$



д) $TREE = \left\{ \begin{matrix} (x_1, x_3), (x_3, x_4), \\ (x_4, x_5) \end{matrix} \right\}$



е) $TREE = \left\{ \begin{matrix} (x_1, x_3), (x_3, x_4), \\ (x_4, x_5), (x_4, x_2) \end{matrix} \right\}$



ж)

Рис. 5.19. **Построение остовного дерева обходом «в глубину»**

MV[*x*]:= -1
ВСЕ_ЦИКЛ;
MV[*a*]:= 0
СТЕК – пустой;
m_v:= 0;
a → СТЕК;
TREE = ∅;

Организуем цикл, пока не пройдены все вершины графа:
ЦИКЛ_ПОКА (*m_v* ≤ *n* - 1)
…
ВСЕ_ЦИКЛ;

Запишем действия, выполняемые в этом цикле. Организуем цикл, пока найдется хотя бы одна вершина смежная с текущей вершиной *a,* метка которой равна -1:

ЦИКЛ_ПОКА (найдется хотя бы одна *x* ∈ *X* смежная с *a*) И (*MV*(*x*)=-1)

Найти вершину *x* ∈ *X*, смежную с *a*, такую, что *MV*(*x*)=-1. Метке вершины присвоить значение *m_v*+1, само значение *m_v* увеличить на единицу. Вершину *x* записать в стек. Ребро *(x, a)* записать в *TREE*. Текущей вершине *a* присвоить значение вершины *x*.

НАЙТИ *x* ∈ *X*, такую, что СОСЕД *(x, a)* И (*MV*(*x*)=-1)
MV(*x*):= *m_v* + 1;
m_v:= *m_v* + 1;
x → СТЕК;
a:= *x*;
TREE:= *TREE* ∪ (*x, a*)
ВСЕ_ЦИКЛ

Если не нашлось вершины с меткой равной –1 и смежной с текущей вершиной *a*, то выполнить обратный обход.

ЕСЛИ СТЕК не пустой ТО
Удалить текущую вершину из стека.
a ← СТЕК;
Прочитать следующую вершину из стека в *a*.
a ← СТЕК;
Сохранить ее в стеке.
a → СТЕК;
ВСЕ_ЕСЛИ
ВСЕ_ЦИКЛ

Запишем алгоритм полностью.

Алгоритм построения остовного дерева графа $G(X, V)$ ***методом обхода «в глубину»***

```
ЦИКЛ (для всех x ∈ X)
        MV[x]:= -1
ВСЕ_ЦИКЛ;
MV[a]:= 0;
СТЕК – пустой;
m_v:= 0;
a → СТЕК;
TREE = ∅;
ЦИКЛ_ПОКА (m_v ≤ n - 1)
  ЦИКЛ_ПОКА (найдется хотя бы одна x ∈ X смежная с a) И (MV[x]:= -1)
   НАЙТИ x ∈ X, такую, что СОСЕД (x, a) И (MV[x]:= -1)
        MV(x):= m_v + 1;
        m_v:= m_v + 1;
        x → СТЕК;
        a:= x;
        TREE:= TREE ∪ (x, a);
  ВСЕ_ЦИКЛ
      ЕСЛИ СТЕК не пустой ТО
        a ← СТЕК;
        a ← СТЕК;
        a → СТЕК;
      ВСЕ_ЕСЛИ
ВСЕ_ЦИКЛ
ВЫВОД TREE;
КОНЕЦ
```

Построение остовного дерева обходом графа «в ширину»

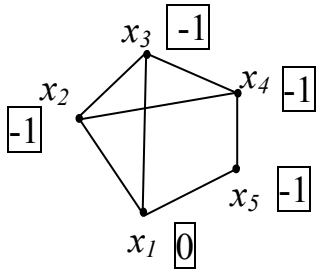
Разберем работу алгоритма на примере построения остовного дерева графа G (рис. 5.18).Начальную вершину обхода выберем произвольно. Пусть это будет вершина x_1. Переменной k присвоим значение $k = 0$. Поставим вершине x_1 метку k. Всем остальным вершинам присвоим метки –1. Вершину x_1 включим во множество $U_k = U_0 = \{x_1\}$ (рис. 5.20-*а*).

Найдем вершины, смежные вершинам из U_k, и ранее не проходимые. Это вершины x_3, x_5, x_2. В дерево TREE запишем ребра (x_1, x_3), (x_1, x_5), (x_1, x_2). Вершинам x_3, x_5, x_2 поставим метки, равные k+1=1.

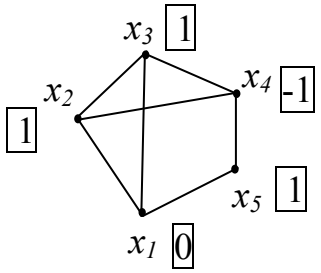
Вершины x_3, x_5, x_2 запишем в $U_{k+1} = U_1 = \{x_3, x_5, x_2\}$. Значение переменной k увеличим на единицу, k:=k+1 (рис. 5.20-*б*).

Найдем вершины, смежные вершинам из множества U_1 и ранее не пройденные. Вершине x_3 смежны вершины x_1, x_2, x_4. Вершины x_1 и x_2 ранее уже проходились (их метки не равны -1). Вершина x_4 ранее не была пройдена (ее метка равна -1). Поставим вершине x_4 метку, равную k+1=2. Запишем вершину x_4 во множество $U_{k+1} = U_2 = \{x_4\}$. Ребро (x_3, x_4) запишем в *TREE*. Вершине x_2 смежны вершины x_3, x_1, x_4. Метки этих вершин не равны –1. Вершине x_5 смежны вершины x_1, x_4. Метки этих вершин не равны –1.

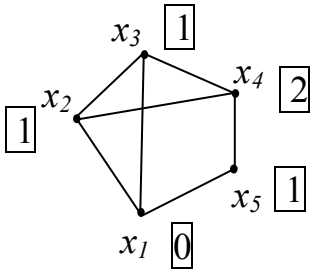
Таким образом, множество $U_{k+1} = U_2 = \{x_4\}$ состоит из одного элемента (рис. 5.20-*в*). После этого алгоритм заканчивает свою работу, т.к. были отмечены все вершины. Построенное остовное дерево представлено на рис. 5.20-*г*.



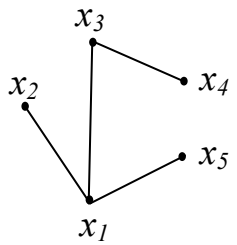
а) $TREE = \varnothing$
k=0, $U_k = \{x_1\}$



б) $TREE = \{(x_1, x_3), (x_1, x_2), (x_1, x_5)\}$
k=1, $U_0 = \{x_1\}$, $U_k = \{x_3, x_2, x_5\}$



в) $TREE = \{(x_1, x_3), (x_1, x_2), (x_1, x_5), (x_3, x_4)\}$
k=2, $U_0 = \{x_1\}$, $U_1 = \{x_3, x_2, x_5\}$,
$U_2 = \{x_4\}$



г)

Рис. 5.20**. Построение остовного дерева методом обхода графа «в ширину»**

Запишем алгоритм построения остовного дерева методом обхода графа «в ширину».

Отметим все вершины графа метками –1. Начальную вершину a отметим меткой 0. Переменной k присвоим значение 0, k:=0. Вершину a запишем во множество U_k. Множество ребер дерева – пустое множество.

ЦИКЛ (для всех $x \in X$)
    $MV[x]$:=-1;
ВСЕ_ЦИКЛ
$MV[a]$:=0;
k:=0;
U_k:= $\{a\}$;
$TREE$:= $\varnothing$;

Организуем цикл, пока найдется хотя бы одна вершина, смежная вершине из U_k, метка которой равна –1:

ЦИКЛ_ПОКА (найдется хотя бы одна вершина $x \in X$, такая что x смежна вершине из U_k и $MV[x]$:=-1)

Множество вершин следующего уровня – пустое множество:
U_{k+1}:= $\varnothing$;

Организуем просмотр всех вершин $x \in U_k$
ЦИКЛ (для $x \in U_k$)

Среди всех вершин графа будем искать вершины, смежные с вершиной из U_k, метки которых равны –1.

ЦИКЛ (для $y \in X$)
ЕСЛИ СОСЕД (x, y) И $MV[y]$:=-1

Изменим метку найденной вершины y. Вершину y запишем в U_{k+1}. Ребро (x, y) запишем в дерево.

ТО $MV[y]$:= k +1;
$y \cup U_{k+1}$;
$(x, y) \cup TREE$;
ВСЕ_ЕСЛИ
ВСЕ_ЦИКЛ
ВСЕ_ЦИКЛ

Увеличим значение переменной k:
k:=k+1;
ВСЕ_ЦИКЛ

Далее запишем алгоритм полностью.

Алгоритм построения остовного дерева способом обхода графа «в ширину».

ЦИКЛ (для всех $x \in X$)
$MV[x]$:= -1;
ВСЕ_ЦИКЛ
$MV[a]$:= 0;
k:= 0;
U_k:= {a};
$TREE$:= ∅;
ЦИКЛ_ПОКА (найдется хотя бы одна вершина $x \in X$, такая что x смежна вершине из U_k и $MV[x]$:= -1)
U_{k+1}:= ∅;
ЦИКЛ (для $x \in U_k$)
ЦИКЛ (для $y \in X$)
ЕСЛИ СОСЕД (x, y) И $MV[y]$:= -1
ТО $MV[y]$:= k + 1;
$y \cup U_{k+1}$;
(x, y) $\cup$ $TREE$;
ВСЕ_ЕСЛИ
ВСЕ_ЦИКЛ
ВСЕ_ЦИКЛ
k:=k + 1;
ВСЕ_ЦИКЛ
КОНЕЦ.

В нагруженных графах существует задача построения минимального остова.

Минимальным остовным деревом нагруженного графа называется остовное дерево, сумма весов ребер которого наименьшая (рис. 5.21).

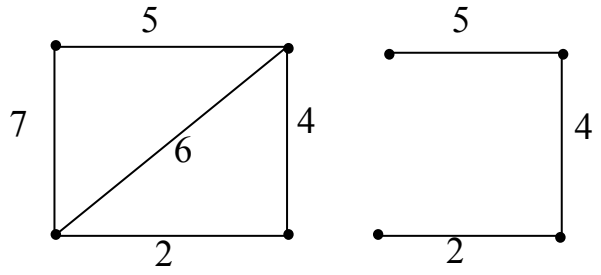


Рис. 5.21. **Граф и его минимальное остовное дерево**

Опишем алгоритм построения минимального остовного дерева на примере графа G (рис. 5.22).

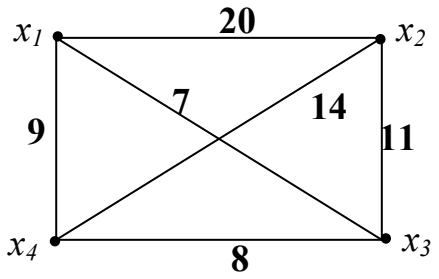


Рис. 5.22. **Нагруженный граф *G***

Выберем в графе G ребро с минимальным весом. Это ребро (x_1, x_3). Запишем это ребро в дерево, $TREE = \{(x_1, x_3)\}$. Концевые вершины ребра x_1 и x_3 запишем во множество вершин дерева $V = \{x_1, x_3\}$ (рис. 5.23-*а*).

Найдем в графе G ребра, инцидентные вершинам из V и не вошедшие в дерево. Это ребра (x_1, x_4), (x_1, x_2), (x_3, x_4), (x_3, x_2).

Среди этих ребер найдем ребро с минимальным весом. Это ребро (x_3, x_4). Запишем его в дерево, $TREE = \{(x_1, x_3), (x_3, x_4)\}$. Вершину x_4 запишем в $V = \{x_1, x_3, x_4\}$ (рис. 5.23-*б*).

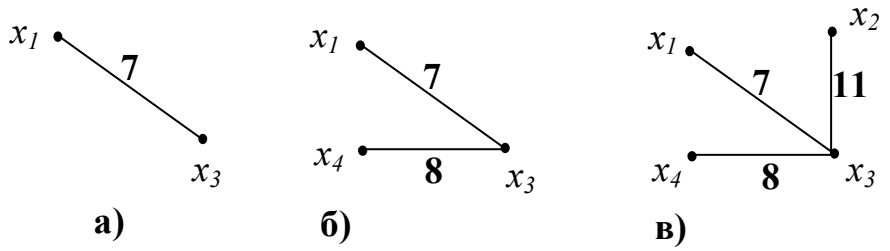


Рис. 5.23. **Построение минимального остовного дерева**

Очевидно, что в графе на трех вершинах уже может существовать цикл. Но в остовном дереве циклов быть не должно. Ребро (x, y), добавляемое в дерево, будет образовывать цикл, если обе его концевые вершины уже принадлежат множеству вершин. Например, ребро (x_1, x_4) не может быть включено в дерево, т.к. его добавление ведет к появлению цикла.

Тогда вершинам из V инцидентны ребра, не образующие цикла и не включенные в дерево – (x_1, x_2), (x_3, x_2), (x_4, x_2). Выберем из них минимальное. Это ребро (x_3, x_2). Запишем его в дерево, $TREE = \{(x_1, x_3), (x_3, x_4), (x_3, x_2)\}$. Вершину x_2 запишем в $V = \{x_1, x_3, x_4, x_2\}$ (рис. 5.23-*в*). Множество вершин дерева V содержит все вершины графа G. Следовательно, остовное дерево построено (рис. 5.23-*в*), алгоритм закончил свою работу.

Пусть граф G задан матрицей весов $W = \begin{bmatrix} \infty & 20 & 7 & 9 \\ 20 & \infty & 11 & 14 \\ 7 & 11 & \infty & 8 \\ 9 & 14 & 8 & \infty \end{bmatrix}$.

Рассмотрим алгоритм построения минимального остовного дерева. Первое ребро, добавляемое в дерево, искалось как ребро с минимальным весом. Тогда в матрице весов найдем минимальный элемент. Это w_{13}=7. Индексы минимального элемента соответствуют концевым вершинам выбранного ребра. Тогда во множество ребер дерева запишем ребро (1,3), а во множество вершин дерева запишем вершины 1 и 3 – $TREE$= {(1, 3)}, V= {1, 3}. Для того, чтобы не рассматривать уже включенные в дерево ребра будем заменять их веса на ∞. Т.к. матрица W - симметричная матрица, то $w_{13} = w_{31} = \infty$.

$$W = \begin{bmatrix} \infty & 20 & \infty & 9 \\ 20 & \infty & 11 & 14 \\ \infty & 11 & \infty & 8 \\ 9 & 14 & 8 & \infty \end{bmatrix}.$$

Для того, чтобы найти ребра, инцидентные вершине k, в матрице весов необходимо просмотреть строку с номером k. Просмотрим строки 1 и 3 в матрице W и найдем среди элементов этих строк минимальный. Это элемент w_{34}= 8. Ребро (3,4) запишем во множество ребер дерева $TREE$= {(1, 3), (3, 4)}, вершину 4 запишем в множество вершин дерева – V= {1, 3, 4}. Заменим вес включенного ребра (3,4) на ∞. Аналогичным образом поступим с ребрами, включение которых в дерево приведет к образованию цикла. Это ребро (1,4), ему соответствует элемент матрицы W $w_{14} = w_{41}$.

$$W = \begin{bmatrix} \infty & 20 & \infty & \infty \\ 20 & \infty & 11 & 14 \\ \infty & 11 & \infty & \infty \\ \infty & 14 & \infty & \infty \end{bmatrix}.$$

Среди элементов строк 1,3,4 найдем минимальный. Это w_{32}= 11. Ребро (3,2) запишем во множество ребер дерева $TREE$= {(1, 3), (3, 4), (3, 2)}, вершину 2 запишем во множество вершин дерева - V= {1, 3, 4, 2}.

Множество вершин дерева содержит все вершины графа G. Построение остовного дерева закончено.

Указание. Для расчета индексов элементов, значения которых заменяются на конкретном шаге алгоритма на ∞, воспользуйтесь декартовым произведением $V \times V$, где V - множество вершин дерева на этом шаге алгоритма.

5.7. Базис независимых циклов

Рассмотрим граф G (рис. 5.24-*а*). Проставим произвольную ориентацию на ребрах графа (рис. 5.24-*б*).

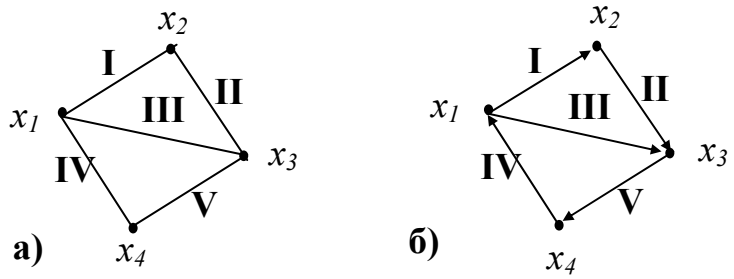


Рис. 5.24. **Неорграф *G***

Рассмотрим цикл в графе G - $\mu_1 = x_1\, x_2\, x_3\, x_1$. При выбранной ориентации ребер будем говорить, что цикл μ_1 проходит ребро (x_1, x_2) ***в прямом направлении***, ребро (x_2, x_3) ***в прямом направлении***, ребро (x_3, x_1) ***в обратном направлении***, т.е. «против» ориентации.

Вектор-циклом $C(\mu)$называется вектор размерности $\overline{1, m}$, элементы которого равны

$$C_i(\mu) = S_i^+(\mu) - S_i^-(\mu),$$

где $S_i^+(\mu)$ - число проходов цикла μ по ребру i в прямом направлении;

$S_i^-(\mu)$ - число проходов цикла μ по ребру i в обратном направлении.

Построим на графе G (рис. 5.24) циклы:

$\mu_1 = x_1\, x_2\, x_3\, x_1$;

$\mu_2 = x_1\, x_2\, x_3\, x_4\, x_1$;

$\mu_3 = x_3\, x_1\, x_4\, x_3$.

Составим для них вектор-циклы:

	I	II	III	IV	V
$C(\mu_1) =$	[1	1	-1	0	0];
$C(\mu_2) =$	[1	1	0	0	0];
$C(\mu_3) =$	[0	0	-1	-1	-1].

Циклы μ_1, μ_2, μ_3 - простые циклы.

Цикл μ считается ***линейной комбинацией циклов*** μ_1, μ_2,…, μ_k, если вектор-цикл $C(\mu)$есть линейная комбинация вектор-циклов $C(\mu_1)$, $C(\mu_2)$,…,$C(\mu_k)$, т.е.:

$$C(\mu) = a_1 \cdot C(\mu_1) + a_2 \cdot C(\mu_2) + \ldots + a_k \cdot C(\mu_k),$$

где $a_1, a_2, \ldots, a_k \in Q$, Q - множество рациональных чисел.

Цикл μ_2 графа G (рис. 5.24) - линейная комбинация циклов μ_1 и μ_3:

$$C(\mu_2) = 1 \cdot C(\mu_1) + (-1) \cdot C(\mu_3).$$

$$1 \cdot [1 \quad 1 \quad -1 \quad 0 \quad 0] + (-1) \cdot [0 \quad 0 \quad -1 \quad -1 \quad -1] = [1 \quad 1 \quad 0 \quad 1 \quad 1].$$

Циклы $\mu_1, \mu_2, \ldots, \mu_k$ называются ***линейно независимыми циклами***, если соответствующие им вектора $C(\mu_1), C(\mu_2), \ldots, C(\mu_k)$ линейно независимы.

Замечание 1. Любой не простой цикл можно представить линейной комбинацией простых циклов.

Рассмотрим цикл $\mu_4 = x_1\, x_2\, x_3\, x_1\, x_4\, x_3\, x_1$ на графе G (рис. 5.24). Соответствующий ему вектор-цикл $C(\mu_4) = [1 \quad 1 \quad -2 \quad -1 \quad -1]$. Действительно, вектор $C(\mu_4)$ - линейная комбинация $C(\mu_3)$ и $C(\mu_1)$:

$$[1 \quad 1 \quad -1 \quad 0 \quad 0] + [0 \quad 0 \quad -1 \quad -1 \quad -1] = [1 \quad 1 \quad -2 \quad -1 \quad -1].$$

Система из линейно независимых циклов с максимальным числом компонентов называется ***базисом из независимых циклов***.

Из *замечания 1* следует, что базис составляют лишь простые циклы.

Количество циклов, входящих в базис определяет *теорема 5.8.*

Теорема 5.8. Количество циклов, составляющих базис графа, равно цикломатическому числу графа.

Замечание 2. В каждом из циклов, входящих в базис, есть ребро, не вошедшее ни в один из остальных циклов базиса.

Замечание 2 дает нам способ построения базиса из независимых циклов.

Алгоритм построения базиса из независимых циклов

Рассмотрим алгоритм построения базиса на примере графа G (рис. 5.24).

Построим произвольное остовное дерево графа G (рис. 5.25-*а*). Добавим к остовному дереву графа G одно из удаленных ребер, например ребро (x_2, x_3).

Так как при добавлении любого ребра в дереве образуется цикл, то выделим произвольно простой цикл и запишем его в базис. Пусть это будет цикл $\mu_1 = x_1\, x_2\, x_3\, x_1$.

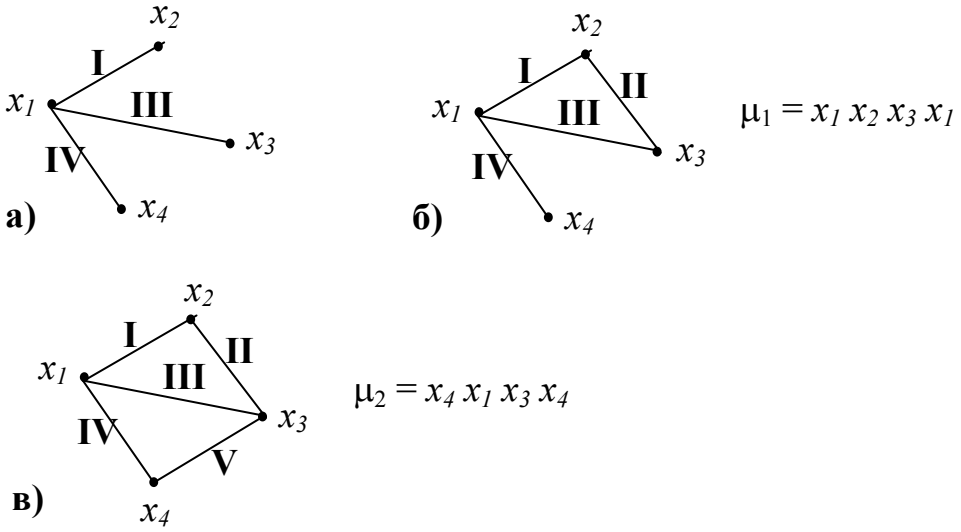


Рис. 5.25. **Построение базиса из независимых циклов**

Добавим следующее удаленное ребро из графа G – ребро (x_4, x_3).

Выделим произвольно в графе простой цикл, который обязательно проходит по добавленному ребру (x_4, x_3), и не проходит по ребру (x_2, x_3).

Пусть это будет цикл $\mu_2 = x_4\ x_1\ x_3\ x_4$. Так как все удаленные ребра из графа G добавлены, то система циклов μ_1, μ_2 - базис из независимых циклов графа G.

Проставим произвольно ориентацию ребер графа G (рис. 5.26) и составим вектор-циклы для μ_1, μ_2 .

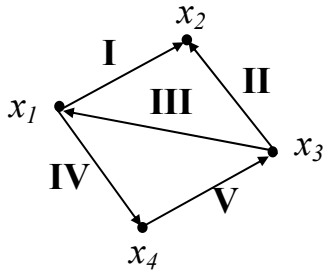


Рис. 5.26. **Построение вектор-циклов базиса**

	I	II	III	IV	V
$C(\mu_1) = [$	1	-1	1	0	0];
$C(\mu_2) = [$	0	0	-1	-1	-1].

Матрица $C(G)$, размерности $\lambda(G) \times m$, составленная из вектор-циклов базиса, называется ***цикломатической матрицей графа*** G.

Для графа G (рис. 5.24) $C(G)=\begin{bmatrix} 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \end{bmatrix}$.

Ранг цикломатической матрицы равен числу строк, т.к. строки матрицы есть линейно независимые вектора.

Выделим в матрице $C(G)$ те столбцы, которые соответствуют удаленным ребрам для получения остова. В графе G удалялись ребра II и V.

$$C(G)=\left[\begin{array}{c|c|cc|c} I & II & III & IV & V \\ \hline 1 & \mathbf{-1} & 1 & 0 & \mathbf{0} \\ 0 & \mathbf{0} & -1 & -1 & \mathbf{-1} \end{array}\right].$$

Из этих столбцов можно построить квадратную диагональную матрицу, элементы главной диагонали которой равны либо единице, либо –1, все остальные элементы равны нулю. Это следует из *замечания 2* и способа построения базиса. Каждый цикл базиса имеет одно ребро, которое не вошло в остальные циклы базиса. Если это ребро j и оно вошло в цикл μ_i, то c_{ij}=1 либо c_{ij}=-1. Во всех остальных строках $c_{kj} = 0$, $k = 1, \lambda(G)$, $k \neq i$.

5.8. Решение задачи 5 контрольной работы №4

Задача 5.

Найти цикломатическое число графа G (рис. 5.27).
Построить остовное дерево графа G (рис. 5.27).
Построить базис из независимых циклов графа G (рис. 5.27).
Проставить произвольную ориентацию ребер графа G.
Построить вектор-циклы для циклов из базиса.
Построить цикломатическую матрицу графа.

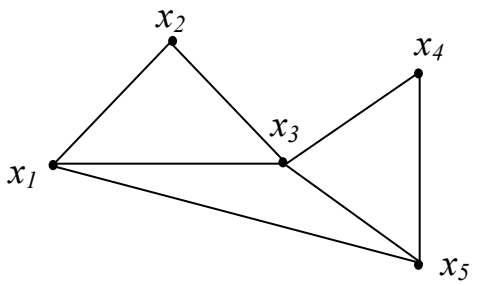


Рис. 5.27. **Граф *G* к задаче 5**

Решение. Пронумеруем вершины графа в произвольном порядке (рис. 5.27). Цикломатическое число графа $\lambda(G) = m - n + k(G)$.

В графе G число вершин - $|X| = 5$, число ребер - $|V| = 7$, граф связный, следовательно, число компонент связности – $k(G) = 1$. Тогда цикломатическое число графа $\lambda(G) = 7 - 5 + 1 = 3$.

Построим остовное дерево графа. Из исходного графа для получения остовного дерева необходимо удалить ровно $\lambda(G) = 3$ ребра.

Построим остовное дерево, используя алгоритм обхода графа «в ширину». Начнем обход с вершины x_1. Поставим вершине x_1 метку 0. Всем остальным вершинам метки –1 (рис. 5.28). Вершине x_1 смежны вершины x_2, x_3, x_5. Меткам вершин x_2, x_3, x_5 присвоим значение 1.

Запишем в дерево ребра (x_1, x_2), (x_1, x_3), (x_1, x_5). Найдем вершины, смежные вершинам x_2, x_3, x_5, метки которых равны -1. Это вершина x_4, смежная вершине x_3. Метке вершины x_4 присвоим значение 2. Ребро (x_3, x_4) запишем в дерево. На этом построение остовного дерева заканчивается, т.к. были отмечены все вершины.

Построенное дерево выделено на рис. 5.28.

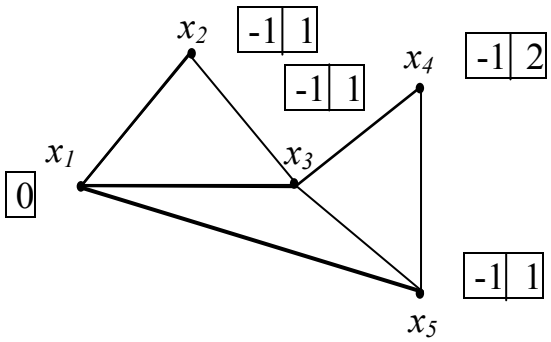


Рис. 5.28. **Остовное дерево графа *G***

Используем построенное остовное дерево для построения базиса из независимых циклов.

По *теореме 5.8.* базис графа G состоит из трех циклов. Построим их. Добавим в остовное дерево графа G ребро (x_2, x_3). Включим это ребро в простой цикл $\mu_1 = x_1 x_2 x_3 x_1$.

Добавим следующее удаленное ребро (x_4, x_5) и построим цикл, в который ребро (x_4, x_5) входит, а ребро (x_2, x_3) не входит. $\mu_2 = x_3 x_4 x_5 x_1 x_3$. Добавим последнее удаленное ребро (x_5, x_3). Построим цикл, в который ребро (x_5, x_3) входит, а ребра (x_2, x_3), (x_4, x_5) не входят. $\mu_3 = x_5 x_3 x_1 x_5$.

Циклы μ_1, μ_2, μ_3 составляют базис из независимых циклов.

Проставим произвольную ориентацию ребер графа G (рис. 5.29). Для выбранной ориентации вектор-циклы, соответствующие циклам μ_1, μ_2, μ_3 :

$$C(\mu_1) = [1 \quad 1 \quad 0 \quad 0 \quad 0 \quad -1 \quad 0]$$
$$C(\mu_2) = [0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad -1]$$
$$C(\mu_2) = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad -1 \quad 1]$$

Составим цикломатическую матрицу графа:

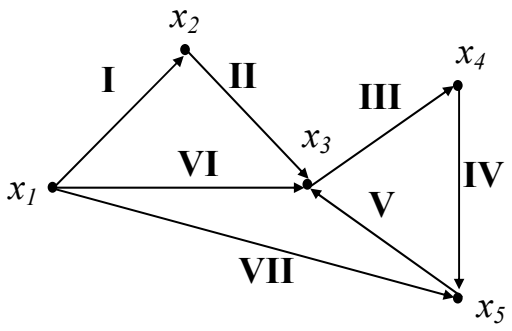


Рис. 5.29. **Ориентация ребер графа *G***

$$C(G) = \begin{array}{c} \begin{array}{ccccccc} \text{I} & \text{II} & \text{III} & \text{IV} & \text{V} & \text{VI} & \text{VII} \end{array} \\ \left[\begin{array}{ccccccc} 1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{array}\right] \end{array}$$

Для получения остовного дерева из графа удалялись ребра II, IV, V. Рассмотрим столбцы II, IV, V в матрице $C(G)$. Из элементов этих столбцов получается диагональная матрица:

$$\begin{array}{c} \begin{array}{ccc} \text{II} & \text{IV} & \text{V} \end{array} \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \end{array}$$

5.9. Контрольные вопросы и упражнения

1. Сколько различных деревьев можно построить на 4 вершинах?
2. Постройте дерево по его коду (1,1,2,2).
3. Сколько вершин в дереве, заданном кодом (2,2,3,4,4).
4. Сколько висячих вершин в дереве, заданном кодом (2,2,3,4,4).

5. Постройте несколько остовных деревьев графа (рис. 5.30)

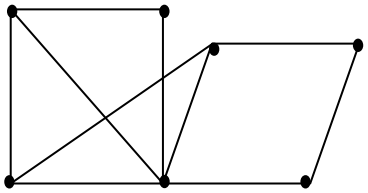


Рис. 5.30. **Неорграф**

6. Постройте минимальное остовное дерево графа, заданного матрицей весов: $W = \begin{bmatrix} \infty & 16 & \infty & 4 & \infty \\ 16 & \infty & 9 & 18 & \infty \\ \infty & 9 & \infty & 22 & 6 \\ 4 & 18 & 22 & \infty & 17 \\ \infty & \infty & 6 & 17 & \infty \end{bmatrix}$.

7. Постройте базис из независимых циклов графа G (рис. 5.30).

8. Постройте вектор-циклы для этих циклов.

9. Чему равен ранг цикломатической матрицы связного графа с $|X| = 7$, $|V| = 13$?

СПИСОК ЛИТЕРАТУРЫ

1. Емеличев В.А.,Мельников О.И... Лекции по теории графов. - М.: Наука, 1990.

2. Липский В. Комбинаторика для программистов. - М.: Мир, 1988. –

3. Нефедов В.Н., Осипова В.А. Курс дискретной математики. - М.: Изд-во МАИ, 1992. – 262 с.

4. Оре О. Теория графов. М: Наука, 1980. – 336 с.

5. Проценко В.С., Чаленко П.И., Сорока Р.А. Техника программирования. - Киев: Высш. шк., 1990. - 184 с.

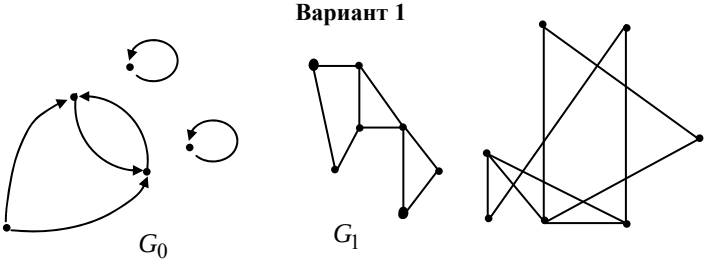
6. Шевелев Ю.П. Высшая математика 6. Дискретная математика. Ч. 2: Теория конечных автоматов. Комбинаторика. Теория графов.: Учебное пособие – Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 1999. – 120 с.

7. Новиков Ф.А. Дискретная математика для программистов. – Спб.: Питер, 2001. – 304 с.:ил.

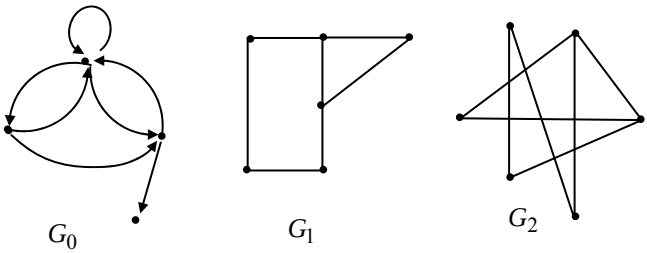
ПРИЛОЖЕНИЕ 1

Графический материал к вариантам контрольных работ.

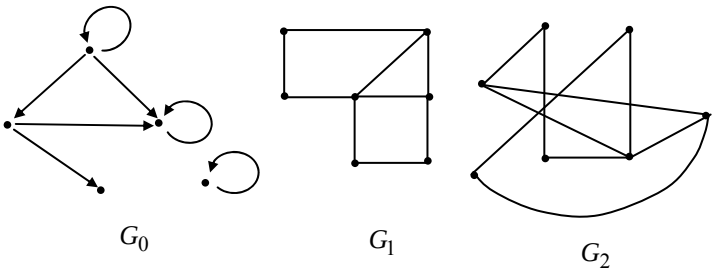
Вариант 1



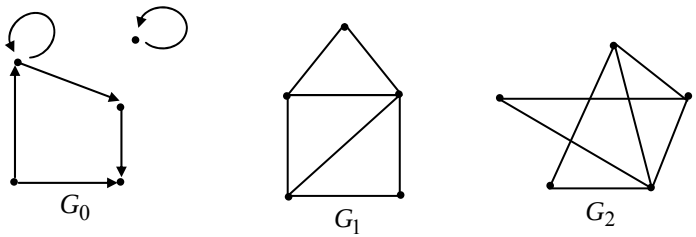
Вариант 2



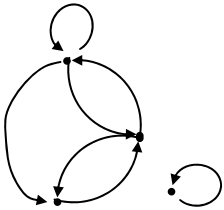
Вариант 3



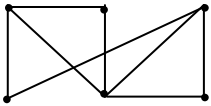
Вариант 4



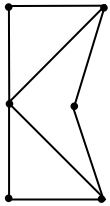
Вариант 5



G_0

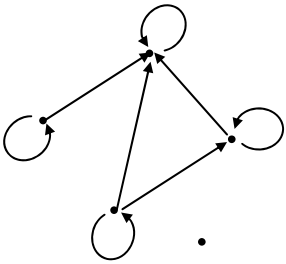


G_1

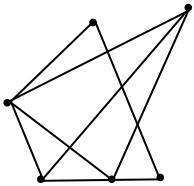


G_2

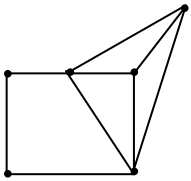
Вариант 6



G_0

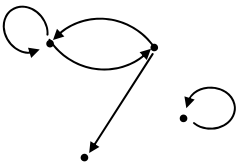


G_1

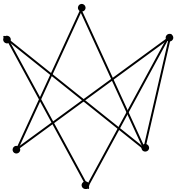


G_2

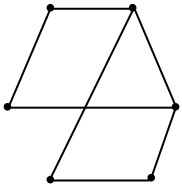
Вариант 7



G_0

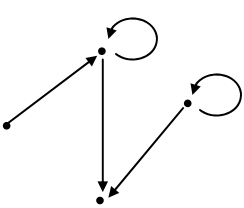


G_1

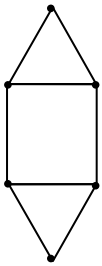


G_2

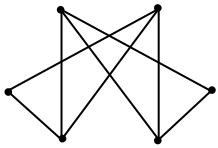
Вариант 8



G_0

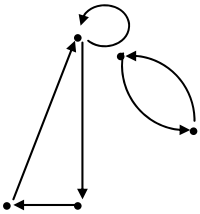


G_1

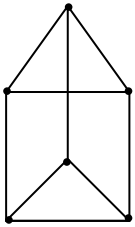


G_2

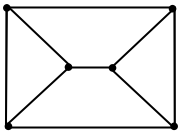
Вариант 9



G_0

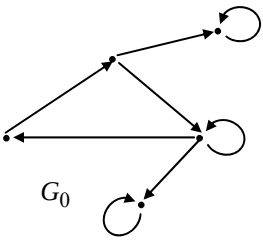


G_1

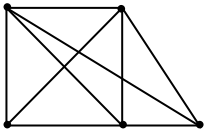


G_2

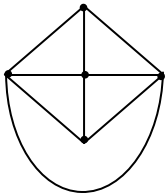
Вариант 10



G_0



G_1



G_2

ПРИЛОЖЕНИЕ 2

Контрольная работа № 3

КОНТРОЛЬНАЯ РАБОТА № 3

Задача 1. Представьте графы G_0 и G_1 различными способами (по четыре для каждого графа).

Задача 2. См. Приложение 3.

Задача 3. Проверьте изоморфность графов G_1 и G_2 ?.

Задача 4. Запишите бинарное отношение, заданное графом G_0. Определите, какими свойствами оно обладает.

Задача 5. Является ли граф G_2 планарным? Если да, то изобразите изоморфный ему плоский граф.

10 вариантов графов (исходные данные для решения контрольных работ № 3 и № 4) находится в Приложении 1.

ПРИЛОЖЕНИЕ 3

Варианты задачи 2 контрольной работы № 3

Вариант 1. Запишите алгоритм и составьте программу нахождения диаметра и радиуса графа по матрице минимальных расстояний.

Вариант 2. Запишите алгоритм и составьте программу получения матрицы смежности орграфа по заданной матрице инцидентности.

Вариант 3. Запишите алгоритм и составьте программу определения вершины графа, имеющей максимальную степень. Граф задан матрицей инцидентности.

Вариант 4. Запишите алгоритм и составьте программу определения степеней всех вершин графа, если граф задан списком ребер.

Вариант 5. Запишите алгоритм и составьте программу которая определяет, является ли бинарное отношение, заданное графом симметричным и рефлексивным. Граф задан матрицей смежности.

Вариант 6. Запишите алгоритм и составьте программу которая вычисляет матрицу инцидентности графа по заданной матрице смежности.

Вариант 7. Запишите алгоритм и составьте программу нахождения вершины графа, имеющей максимальную степень. Граф задан списком ребер.

Вариант 8. Запишите алгоритм и составьте программу нахождения вершины графа, имеющей нулевую полустепень захода. Граф задан матрицей инцидентности.

Вариант 9. Запишите алгоритм и составьте программу выделения компонент связности по матрице смежности.

Вариант 10. Запишите алгоритм и составьте программу вычисления матрицы инцидентности орграфа по заданному списку ребер.

ПРИЛОЖЕНИЕ 4

Контрольная работа № 4

КОНТРОЛЬНАЯ РАБОТА № 4

Задача 1. Запишите матрицы достижимости и взаимодостижимости для графа G_0. Выделите сильные компоненты графа.

Задача 2. Найдите диаметр, радиус, центры графа G_1.

Задача 3. Возможно ли нарисовать граф G_2 не отрывая руки от бумаги? Обоснуйте ответ. Если возможно, запишите произвольный эйлеров цикл или цепь.

Задача 4. См. Приложение 5.

Задача 5. Найдите цикломатическое число графа G_2. Постройте остовное дерево графа G_2.Постройте базис из независимых циклов графа G_2. Проставьте произвольную ориентацию ребер графа и постройте вектор-циклы для циклов из базиса. Постройте цикломатическую матрицу графа G_2.

ПРИЛОЖЕНИЕ 5

Варианты задачи 4 контрольной работы № 4

Вариант 1. Продемонстрируйте волновой алгоритм нахождения кратчайшего пути между выделенными вершинами на графе G_1.

Вариант 2. Продемонстрируйте алгоритм построения остовного дерева графа G_1 методом обхода «в глубину».Обход начните с вершины, имеющей максимальную степень.

Вариант 3. Продемонстрируйте алгоритм построения остовного дерева графа G_2 методом обхода «в ширину».

Вариант 4. Продемонстрируйте алгоритм Дейкстры нахождения кратчайшего пути между вершинами 1 и 5, если известна матрица весов

$$W = \begin{bmatrix} \infty & 3 & 7 & \infty & 20 \\ 3 & \infty & 2 & \infty & \infty \\ 7 & 2 & \infty & 4 & 18 \\ \infty & \infty & 4 & \infty & 1 \\ 20 & \infty & 18 & 1 & \infty \end{bmatrix}$$ (постройте граф по матрице весов).

Вариант 5. Продемонстрируйте алгоритм построения минимального остовного дерева графа, если известна матрица весов

$$W = \begin{bmatrix} \infty & 3 & 7 & \infty & 20 \\ 3 & \infty & 2 & \infty & \infty \\ 7 & 2 & \infty & 4 & 18 \\ \infty & \infty & 4 & \infty & 1 \\ 20 & \infty & 18 & 1 & \infty \end{bmatrix}$$ (постройте граф по матрице весов).

Вариант 6. Продемонстрируйте алгоритм обхода «в глубину» на графе G_1.

Вариант 7. Продемонстрируйте алгоритм Форда для нахождения кратчайшего пути из вершины 1 в вершину 3,на графе, матрица весов которого:

$$W = \begin{bmatrix} \infty & 5 & 17 & 8 & 3 \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 4 & \infty & \infty & \infty \\ \infty & \infty & 5 & \infty & \infty \\ \infty & \infty & \infty & 2 & \infty \end{bmatrix}$$ (нарисуйте граф по матрице весов).

Вариант 8. Продемонстрируйте алгоритм обхода дерева «в ширину». Дерево постройте по заданному коду (3,4,4,5,6,6,6).

Вариант 9. Продемонстрируйте алгоритм обхода дерева «в глубину». Дерево постройте по заданному коду (3,3,4,5,7,6,6).

Вариант 10.Продемонстрируйте алгоритм выделения сильных компонент графа G_0 по матрице взаимодостижимости.