Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании (КСУП)

Е.Ф. Жигалова

ДИСКРЕТНАЯ МАТЕМАТИКА

Часть 2

Учебное методическое пособие

2004

Корректор: Осипова Е.А.

Жигалова Е.Ф.
Дискретная математика. Часть 2: Учебное методическое пособие. – Томск: Томский межвузовский центр дистанционного образования, 2004. – 94 с.

В пособии на общей теоретической основе изложены основные понятия и определения теории графов и комбинаторики, рассмотрены математический аппарат, постановки задач и универсальные методы их решения.

К пособию прилагаются электронные версии обучающих программ базовых алгоритмов задач теории графов.

Пособие рассчитано на студентов технических университетов.

СОДЕРЖАНИЕ

КОНТРОЛЬНАЯ РАБОТА №1

Контрольная работа №1 содержит 8 заданий.

Для выполнения контрольной работы необходимо выбрать свой вариант, пользуясь общим правилом вычисления его номера N (N= K div100, где K – число, образованное последними двумя цифрами пароля студента).

Задание 1.

Для графа ***G=(X,U)*** (рисунок 1) выполнить следующее:

1.1. Построить:

- матрицу смежности;

- матрицу инциденций.

1.2. Определить степени $S(x_i)$ для всех вершин $\{x_i\}$ данного графа. (Указать каким способом вычисляли $S(x_i)$).

1.3. а). Подсчитать количество маршрутов $\mu_{i,j}$ длиной $l=3$ в графе ***G=(X,U)*** для каждой пары вершин.

б). Построить все $\mu_{i,j}$ длиной $l=3$, связывающие вершины x_i и x_k (помечены *****).'

Маршруты записать в форме: $\mu_{ij}^{(p)}$ =(x_i ,… x_t ,…, x_k), где р – номер маршрута. Указать ребра, через которые проходят маршруты.

Примечание. Для выполнения п.1.3а) составить программу на алгоритмическом языке Паскаль (к отчёту приложить исходный код программы и exe-file).

Задание 2.

По матрицам А (рисунок 2) и С (рисунок 3) построить графы *G1* и *G2.*

Задание 3.

Для графа ***G=(X,U)*** (рисунок 1) построить минимальные маршруты, связывающие вершину, помеченную * (любую из двух), с остальными вершинами, указать их длину. Описать способ решения данной задачи.

Задание 4.

Для графа, представленного на рисунке 1 выполнить следующее:

4.1. Привести примеры подграфов 3-х вершинных, 4-х вершинных, 1-вершинных.

4.2. Привести пример суграфа данного графа.

4.3. Выполнить унарные операции для вершин, помеченных *.

Задание 5.

Для графа ***G=(X,U)*** (рисунок 1) выполнить следующее:

5.1. Построить матрицу метрики (отклонений).

5.2. Вычислить радиус и диаметр.

5.3. Определить периферийные точки.

Задание 6.

Произвести произвольно ориентацию рёбер графа ***G=(X,U)*** (рисунок 1) и для нового графа G^N выполнить задания 1.1, 1.3, 5.

Задание 7.

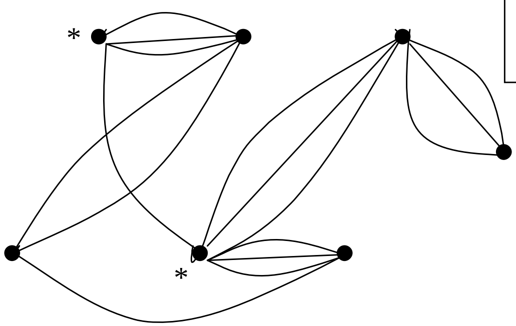
Построить скелет графа G^N.

Задание 8.

В графе ***G=(X,U)*** (рисунок 1) найти все максимальные полные и максимальные пустые подграфы с помощью алгоритма Магу-Уэйсмана.

ВНИМАНИЕ ! Отчёт по контрольной работе №1 выполнять в текстовом редакторе WORD.

Вариант 1



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

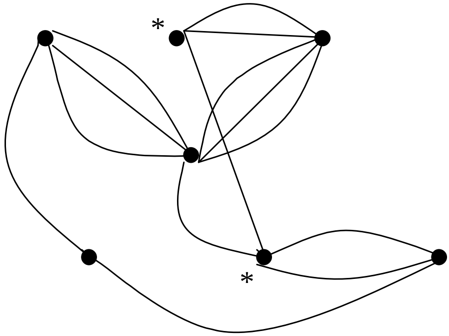
	1	2	3	4	5	6	7
1	0	0	0	0	0	3	0
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	0	1	1	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 2



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

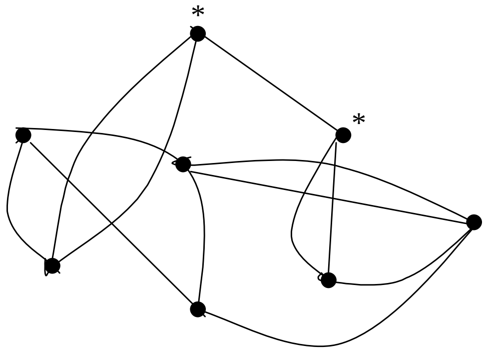
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	0	0	0	0	3	2
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	3	0	0	1	0	0	1
7	2	0	0	0	0	1	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	0	1	0	0	0	1
2	1	1	0	1	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 3



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

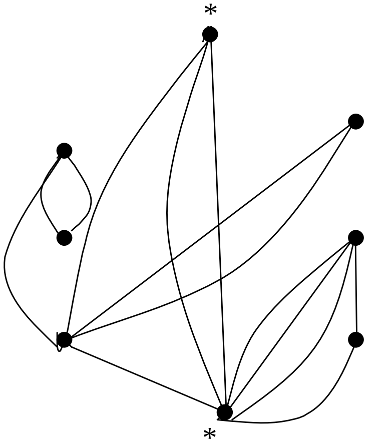
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	0	0	0	0	4	0
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	4	0	0	1	0	0	2
7	0	0	0	0	0	2	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	0	0	1	1	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 4



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

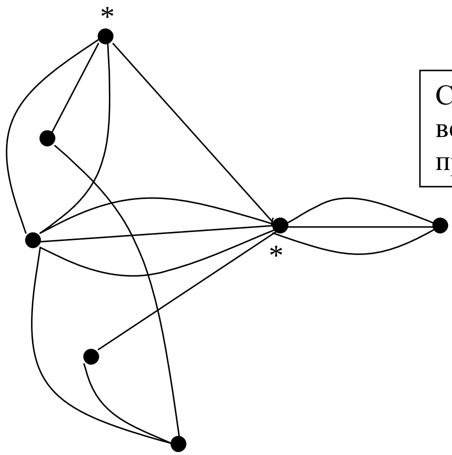
	1	2	3	4	5	6	7
1	0	0	0	0	3	0	0
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	3	1	0
5	3	0	0	3	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	0	1	1	0	0	0	0
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 5



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

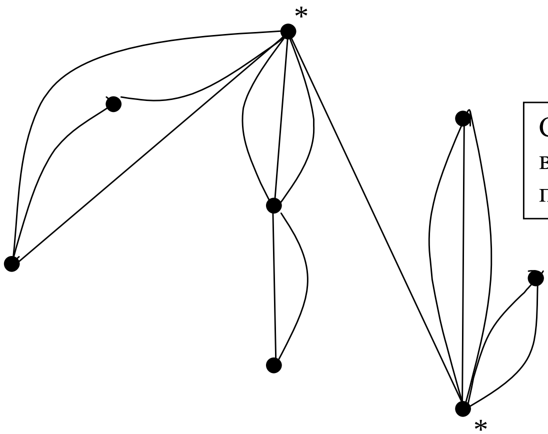
	1	2	3	4	5	6	7
1	0	0	0	0	0	3	0
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	2	1	0
5	0	0	0	2	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	0	1	1	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	1	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	0	1	0

Рисунок 3

Вариант 6.



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

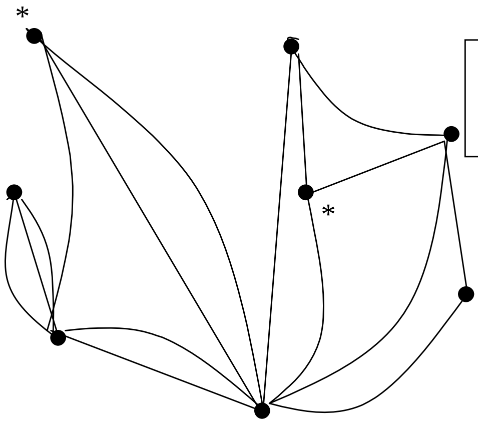
	1	2	3	4	5	6	7
1	0	0	0	0	0	3	0
2	0	0	0	0	1	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	1	0	1	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	0	1	1	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	1	0	0	1	1
4	1	0	0	0	0	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 7



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

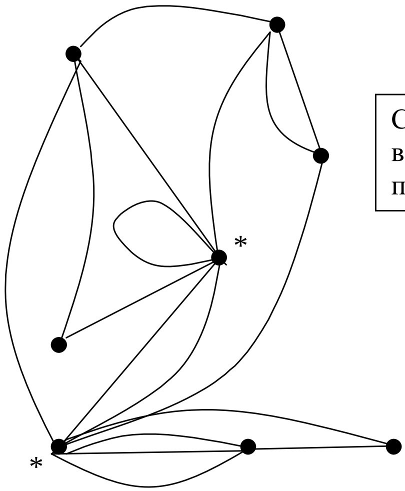
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	2	0	0	0	3	0
2	2	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	1	1	0	0	0	1
2	1	1	0	0	0	1	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	0	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 8



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

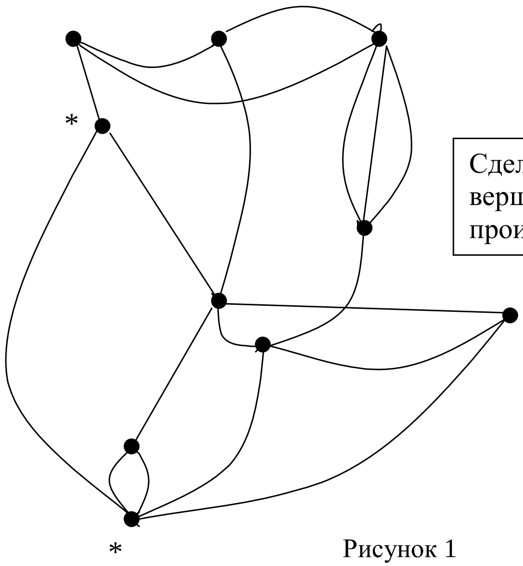
	1	2	3	4	5	6	7
1	2	0	0	0	0	3	0
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	0	1	1	0	0	0	1
2	1	0	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 9



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

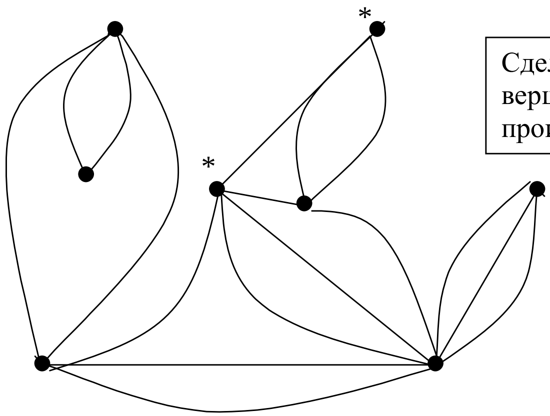
	1	2	3	4	5	6	7
1	0	0	0	0	0	3	0
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	1
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	0	1	0	0	0	0	1
2	1	1	0	0	1	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	0	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 10



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

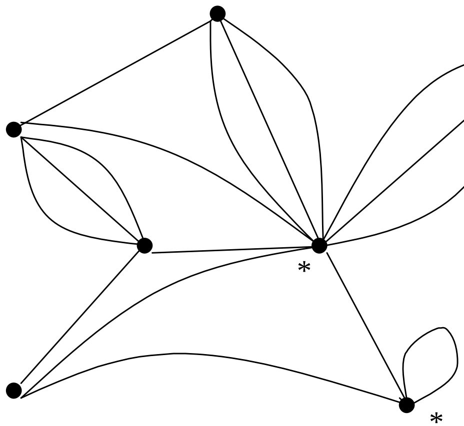
	1	2	3	4	5	6	7
1	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	1	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	0	1	1	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 11



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

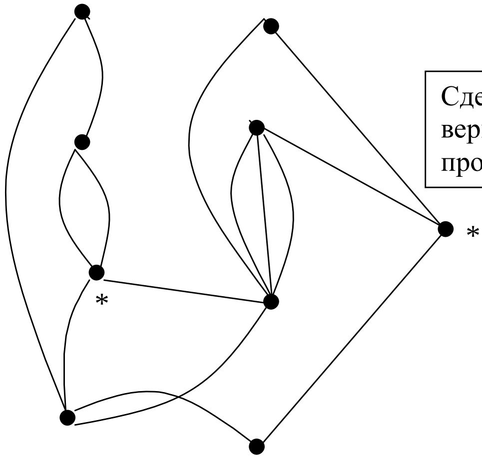
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	0	0	0	0	3	0
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	1	1	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	0	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 12



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

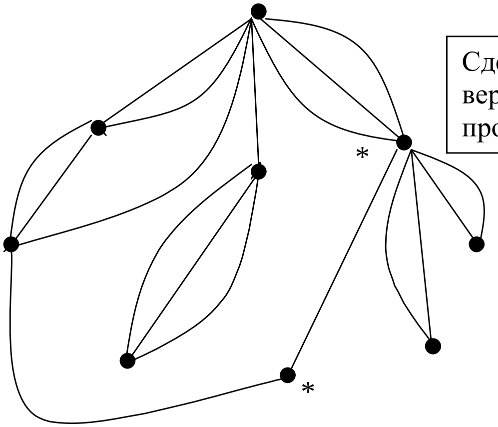
	1	2	3	4	5	6	7
1	0	0	0	0	0	3	0
2	0	0	2	0	0	0	0
3	0	2	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	1	1	1	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 13



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

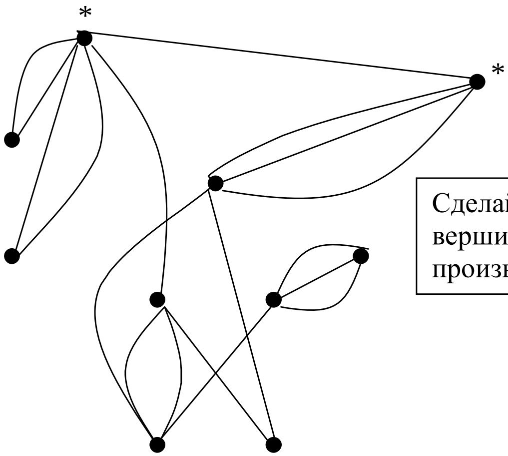
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	0	0	0	0	3	0
2	0	0	0	0	3	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	3	0	1	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	0	0	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	1	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 14



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

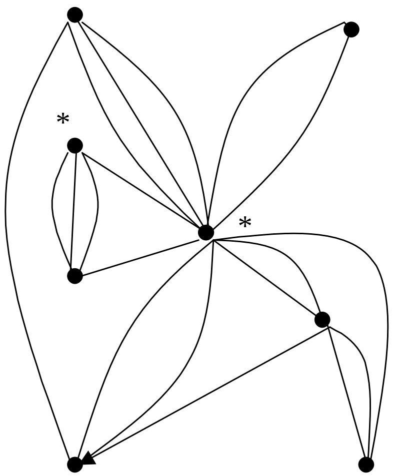
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	0	0	0	0	3	0
2	0	0	0	0	0	3	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	3	3	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	1	1	0	0	0	1
2	1	0	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	1	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 15



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

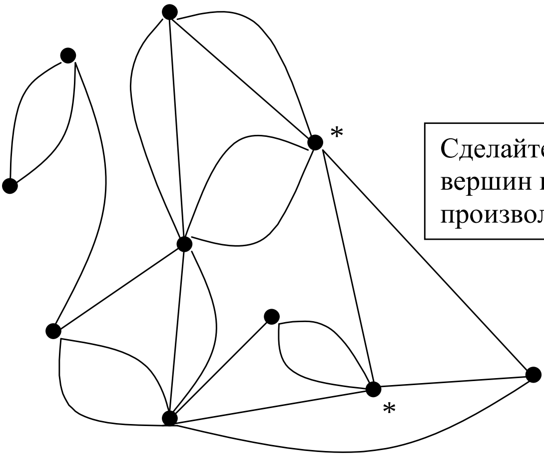
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	2	0	0	0	3	0
2	2	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	3	0	0	1	0	0	4
7	0	0	0	0	0	4	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	1	0	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	1	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 16



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

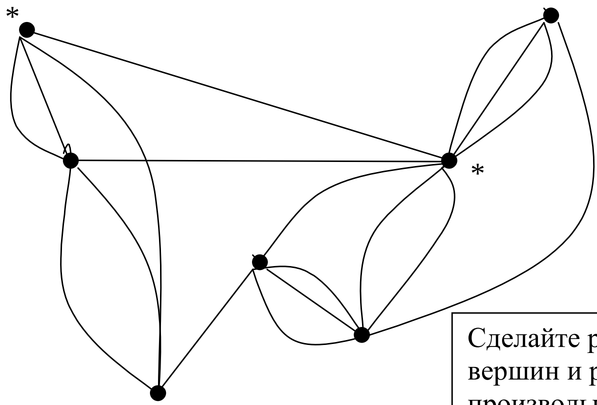
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	0	0	0	0	1	0
2	0	0	2	0	0	0	0
3	0	2	0	2	0	0	0
4	0	0	2	0	1	1	0
5	0	0	0	1	0	0	0
6	1	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	1	1	0	0	0	0
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	1

Рисунок 3

Вариант 17



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

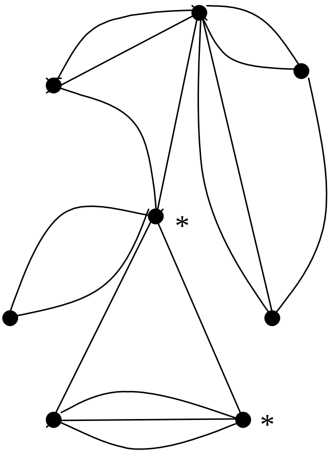
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	0	0	0	0	3	0
2	0	0	0	3	0	0	0
3	0	0	0	0	0	0	0
4	0	3	0	0	1	1	0
5	0	0	0	1	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	1	1	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 18



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

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

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	1	1	0

Рисунок 3

Вариант 19

Сделайте разметку вершин и ребер графа в произвольном порядке

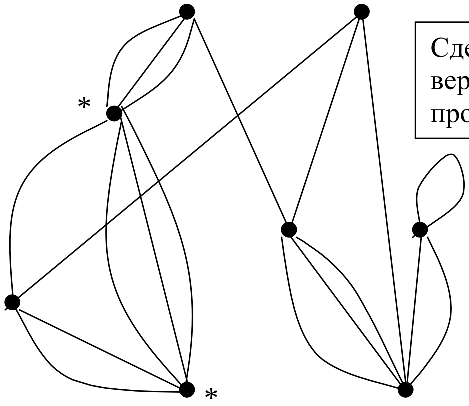


Рисунок 1

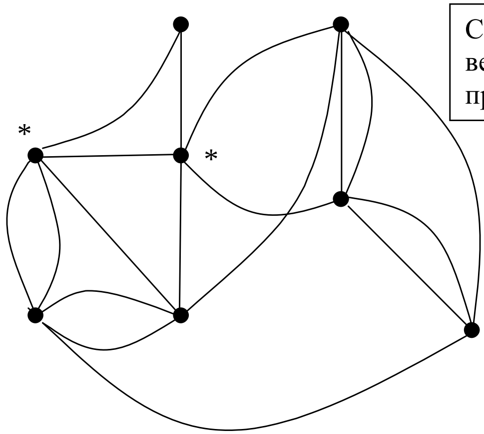
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	0	0	0	0	3	0
2	0	0	0	0	0	0	0
3	0	0	0	2	0	0	0
4	0	0	2	0	0	1	0
5	0	0	0	0	0	0	0
6	3	0	0	1	0	0	1
7	0	0	0	0	0	1	0

Рисунок 2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	1	1	0	0	0	1
2	1	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	1	1
4	1	0	0	0	1	1	0	0	0
5	0	0	1	0	0	1	1	0	0
6	0	0	0	0	0	0	0	0	0

Рисунок 3

Вариант 20



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

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

Рисунок 2

	1	2	3	4	5	6	7	8	9
1	0	1	0	1	1	0	0	0	0
2	1	1	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	1	0
4	1	0	0	0	0	1	0	0	0
5	0	0	1	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	1

Рисунок 3

<u>Тема 1.</u> ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ТЕОРИИ ГРАФОВ

1.1 Определение графа

Теория графов оперирует формальными моделями объектов, имеет дело со свойствами самих графов независимо от того, какова природа объектов, описываемых графами. Использование аппарата теории графов для разработки алгоритмов конструкторского и технологического проектирования схем ЭВА приводит к повышению эффективности и качества создаваемых объектов.

Понятие графа опирается на понятие множества. Графом можно назвать объект, состоящий из двух множеств: множества точек и множества линий. Множество точек графа обозначается $X=\{x_1,x_2,\dots,x_n\}$ и называется множеством вершин. Суммарное число n всех вершин графа называется мощностью множества X графа и обозначается $|X|=n$.

Множество линий, соединяющих любые пары вершин (x_i, x_j), принадлежащих множеству X, называется множеством ребер или дуг и обозначается:

$U=\{u_1, u_2,\dots,u_m\}$, где u_i – ребро графа.

Суммарное число m всех рёбер графа называется мощностью множества рёбер графа и обозначается $|U|=m$.

Таким образом, графом можно считать математический объект, который обозначается $G=(X,U)$ и состоит из множества вершин X и множества ребер U, находящихся между собой в некотором отношении.

В общем случае множество U можно представить в виде

$$U=\bar{U}\bigcup \overset{o}{U}\bigcup \vec{U}.$$

$\overline{U}$ – подмножество неориентированных линий, для которых не существенен порядок соединения вершин. Подмножество $\overline{U}$ называется подмножеством ребер. Причем каждое ребро

$u_i \in \overline{U}$ определяется неупорядоченной парой вершин x, y, которые оно соединяет, и записывается: $u_i=(x, y)$ или $u_i=(y, x)$.

$\vec{U}$ – подмножество ориентированных линий, для которых существенен порядок соединения вершин. Подмножество $\vec{U}$ называется подмножеством дуг. Причем каждая дуга $u_i \in \vec{U}$ определяется упорядоченной парой (кортежем длины два) вершин x_k, y_l, которые u_i соединяет, и записывается: $u_i=(x_k,y_l)$.

Заметим, что $u_i=(x_k,y_l)$ и $u_j=(u_l,x_k)$ – это различные дуги в графе G;

$\overset{o}{U}$ – подмножество линий, каждая из которых выходит и входит в одну и ту же соответствующую этой линии вершину. Называется $\overset{o}{U}$ подмножеством петель.

Каждая петля u_i принадлежит множеству $\overset{o}{U}$ и может определяться упорядоченной парой, например вида $u_i=(x_k,x_k)$.

Граф состоит из вершин, которые на плоскости изображаются нумерованными кружками или точками, и рёбер, изображаемых линиями (со стрелками или без стрелок), которые соединяют некоторые из этих вершин. Однонаправленное соединение ребром двух вершин называется дугой. Двунаправленные или ненаправленные рёбра называются звеньями. Рёбра, соединяющие вершину саму с собой, называются петлями.

Рёбра, подходящие к вершине x, называются *инцидентными* данной вершине. Соответственно говорят, что вершина x *инцидентна* рёбрам, подходящим к ней.

Количество рёбер, инцидентных вершине x, называется *степенью* вершины s(x).

Вершина x называется изолированной, если её степень s(x) равна нулю.

Граф является конечным, если он имеет конечное число вершин и рёбер.

Бесконечные графы обладают следующими характеристиками:

1. Вершинами графа служат натуральные числа, причём вершины p и q соединены звеном в том и только том случае, если оба числа p и q простые и $| p–q | \le 2$. Множество вершин этого графа счётно, а является ли множество рёбер счётным или только конечным – неизвестно до сих пор (проблема близнецов в теории чисел).

2. Вершинами являются числа 1,2,...,n, а каждое действительное число x, удовлетворяющее условию $i < x < i+1$, служит дугой из вершины i в вершину i+1. Граф содержит конечное множество вершин и континуум рёбер (дуг).

3. Вершинами служат все действительные числа, и при фиксифованном $\delta > 0$ вершины x и y соединены ребром (звеном или петлёй) тогда и только тогда, когда $| x–y | < \delta$. Каждому значению δ отвечает свой граф, у которого множества вершин и рёбер оба континуальны.

1.2 Классы графов

Класс орграфов (ориентированных графов). Это граф G=(X,U), у которого $\overline{U} = \varnothing$.

Класс неорграфов (неориентированных графов). Это граф G=(X,U), у которого $\vec{U} = \varnothing$.

Класс смешанных графов. Это граф G=(X,U), у которого

$\overline{U} \subset U$, $\vec{U} \subset U$ и $\overline{U} \cup \vec{U} \subseteq U$.

Класс мультиграфов. Мультиграф – это граф G=(X,U), у которого имеются параллельные (кратные) рёбра, т.е.

$\exists\, x,y \in X \mid x\, u_k\, y$, $x\, u_m\, y, \ldots, x\, u_p\, y$, $u_k, u_m, \ldots, u_p \in U$.

Для некоторых классов графов естественно определяется понятие *полного графа*, как такого, который содержит все рёбра, возможные при принадлежности графа данному классу и при неизменном множестве вершин. Например, в случае *p-графа* полнота означает, что при каждой вершине имеется ровно *p* петель, а каждая пара различных вершин связана ровно *p* рёбрами (среди них могут быть как звенья, так и дуги).

Граф общего вида, в котором две различные вершины всегда смежны, называется *плотным*.

Особо важную роль играют так называемые *обыкновенные* графы. Граф этого класса характеризуется следующими четырьмя свойствами:

1) он конечен;

2) он является неориентированным, т.е. не содержит дуг;

3) он не содержит петель;

4) он не содержит «параллельных» («кратных») рёбер; иначе говоря, никакие две его вершины не могут соединяться более чем одним ребром (звеном).

Определение:

Обыкновенный граф – это неориентированный униграф без петель. *Униграф* – это граф, в котором смежные вершины связаны только одним неориентированным ребром.

Дополнительный граф. Дополнительным графом для обыкновенного графа $L=(X,U)$ называется обыкновенный граф $L^*=(X,U^*)$ с тем же множеством вершин X и с множеством $U^* \doteq \{x\tilde{y}/\, x,y \in X \,\&\, x \neq y\} \setminus U$ рёбер. Иначе говоря, это такой граф, в котором две различные вершины смежны тогда и только тогда, когда они не смежны в исходном графе L.

Рассмотрим ещё одно важное в теории графов понятие– *скелет графа*.

Скелет графа общего вида. В случае, когда при исследовании графа L=(X.U;P) общего вида требуется не полная информация о нём, а лишь знание того, какие пары его различных вершин смежны и какие нет, прибегают к носителю такой информации – *скелету* графа L, который обозначим как $\overline{L}=(X,\overline{U})$. Граф $\overline{L}$ относится к классу обыкновенных графов с множеством вершин тем же, что и в графе L, и новым множеством рёбер $\overline{U}$, определённым следующим образом:

1) если в графе L есть петли, то они удаляются;

2) если в графе L есть дуги, то производится дезориентация дуг;

3) если в графе L есть кратные рёбра, то они заменяются одним эквивалентным ребром-звеном;

4) оставшиеся рёбра образуют множество рёбер $\overline{U}$.

Таким образом, множество рёбер $\overline{U}$ состоит из рёбер, полученных из множества U после выполнения описанных выше процедур 1, 2, 3.

1.3 Способы задания графов

1.3.1 Геометрический

Основой геометрического способа задания графа является рисунок, дающий визуальное изображение графа. Изображение графа в виде рисунка наглядно раскрывает содержательный смысл представляемого объекта. В этом способе вершины графа изображаются точками (кружками), а рёбра – линиями (со стрелками или без стрелок), концы которых подходят к вершинам графа.

1.3.2 Описание графа через предикат (инцидентор) P

Говорят, что задан граф G=(X,U,P), если дано множество вершин X, множество ребер U и трёхместный предикат (инцидентор) P, определяющий, какую пару вершин x_i, x_j, принадлежащих множеству вершин X, соединяет ребро $u_k=(x_i,x_j)$. Инцидентор P удовлетворяет двум условиям:

А. Инцидентор P определён на всех таких упорядоченных тройках элементов x, u, y, для которых $x, y \in X$ и $u \in U$.

Б. $\forall u \; \exists x, y \; \{P(x, u, y) \; \& \; \forall x', y' [P(x', u, y') \Rightarrow (x = x' \; \& \; y = y') \vee (x = y' \; \& \; y = x')]\}$.

1.3.3 Матричный

1.3.3.1 Большинство задач автоматизации конструирования схем удобно решать при использовании матричного способа представления графов. Квадратная таблица $R=//r_{i,j}//_{n*n}$ называется *матрицей смежности*, если её строки и столбцы соответствуют вершинам графа, а элементы $r_{i,j}$ образуются по правилу:

- $r_{i,j} = 1$, если вершина x_i соединена с вершиной x_j ребром, т.е. x_i смежна x_j;
- $r_{i,j} = 0$ в противном случае.

Заметим, что для мультиграфа и смешанного графа задают:

- $r_{i,j} = p$, если вершина x_i соединена с вершиной x_j p – числом рёбер;
- $r_{i,j} = 0$ в противном случае.

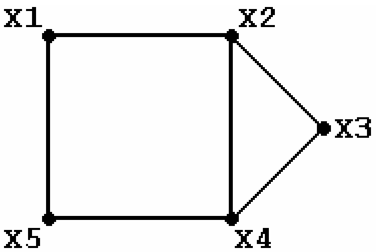


Рисунок 1

Очевидно, что для неорграфов $r_{i,j} = r_{j,i}$ и для их задания можно использовать треугольную матрицу R. Так, для графа на рисунке 1 матрица смежности R имеет вид:

R =

	X1	X2	X3	X4	X5
X1	0	1	0	0	1
X2	1	0	1	1	0
X3	0	1	0	1	0
X4	0	1	1	0	1
X5	1	0	0	1	0

Треугольная матрица R’ для этого же графа запишется

$R' =$

	X1	X2	X3	X4	X5
X1	0	1	0	0	1
X2		0	1	1	0
X3			0	1	0
X4				0	1
X5					0

Строки и столбцы матрицы R соответствуют вершинам графа. На пересечении i-й строки и j-го столбца ставится элемент $r_{i,j}$, соответствующий числу рёбер, соединяющих вершины x_i и x_j. Строки и столбцы матрицы R можно нумеровать числами натурального ряда, соответствующими индексам помеченных вершин. Петлям в графе соответствуют элементы $r_{i,i}$ главной диагонали матрицы R. Преимущесво использования матриц смежности – это простота выполнения преобразований и операций над графами как для конструктора, так и для ЭВМ.

1.3.3.2 Граф можно задавать также матрицей *инциденций* B, строки которой соответствуют вершинам графа, столбцы – ребрам. Элементы $b_{i,j}$ матрицы B для неорграфа могут принимать значения (0 или 1):

$b_{i,j} = 1$, если ребро u_k инцидентно вершине x_i;

$b_{i,j} = 0$, если ребро u_k неинцидентно вершине x_i.

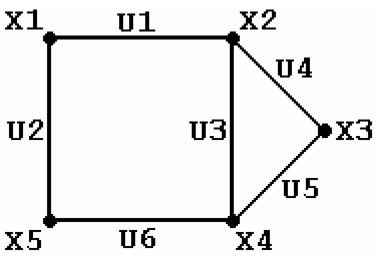


Рисунок 2

Для графа, представленного на рисунке 2, матрица инциденций B имеет вид:

	U1	U2	U3	U4	U5	U6
X1	1	1	0	0	0	0
X2	1	0	1	1	0	0
X3	0	0	0	1	1	0
X4	0	0	1	0	1	1
X5	0	1	0	0	0	1

1.4 Числовые характеристики вершин графа

Каждая вершина графа имеет числовую характеристику s(x), которая называется степенью, или валентностью, вершины. Степень вершины $S(x_i)$ это целое положительное число, равное количеству ребер, инцидентных вершине x_i.

Для ориентированных графрв различают «*полустепени исхода*» и «*полустепени захода*». Это тоже числовые характеристики вершин, соответственно равные количеству дуг, исходящих из вершины и входящих в вершину.

1.5 Маршруты, цепи и циклы

В этой главе будут рассмотрены такие свойства графов, которые не меняются при произвольной ориентации звеньев графа, переориентации или дезориентации дуг (всех или некоторых). Мы рассмотрим такие свойства графов общего вида $L=(X,U;P)$, которые полностью определяются предикатом $\tilde{P}(x,u,y)\dot{\Leftrightarrow} P(x,u,y)\vee P(y,u,x)$, называемым полуинцидентором (неоринцидентором) графа *L*.

О неорграфе $\tilde{L}=(X,U;\tilde{P})$ можно сказать, что он получен из *L* посредством дезориентации его дуг. В свою очередь *L* можно получить из $\tilde{L}$ ориентацией звеньев.

Определение:

Конечная последовательность M элементов графа L**:**

$M = x_0\ u_1\ x_1\ u_2\ x_2 \ldots x_{k-1}\ u_k\ x_k$ ($k >= 0$),

для которой истинно высказывание**:**

$P'(x_0,u_1,x_1)\ \&\ P'(x_1,u_2,x_2)\ \&\ \ldots\ \&\ P'(x_{k-1},u_k,x_k)$, называется маршрутом из вершины x_0 в вершину x_k, или маршрутом, соединяющим вершину x_0 с вершиной x_k; в случае $x_0=x_k$ имеем циклический маршрут при вершине x_0, или цикл. Число k носит название длины маршрута M. Иными словами, длина маршрута равняется числу ребер, входящих в него. Заметим, что маршрут – это не просто часть графа, ибо порядок его обхода играет существенную роль; так, маршрут M1:

$M1 = x_k\ u_k\ x_{k-1} \ldots x_2\ u_2\ x_1\ u_1\ x_0$ при $k>=0$

не совпадает с написанным выше маршрутом M, хотя и состоит из тех же самых элементов и с тем же отношением инцидентности.

Маршрут M:

$M = x_0\ u_1\ x_1\ u_2\ x_2 \ldots x_{k-1}\ u_k\ x_k$

называется цепью, если ребра u_1, u_2,...,u_k различны (при $k<=1$ это условие выполнено «в силу ложности посылки»). Цепь, в случае если $x_0 = x_k$, при $k >= 1$ называется циклом.

Цепь называется простой, если все ее вершины x[p],x[k],...,x[l] различны.

В случае, когда x[0]=x[l] & l>=1, имеем простой цикл, который, будучи цепью, в то же время не есть простая цепь. Цепь, в которой начальная и конечная вершины не совпадают, но есть повторяющиеся вершины, называется циклической.

ЛЕММА.

Всякий маршрут (в частности, всякая цепь) графа содержит хотя бы одну простую цепь, соединяющую ту же пару вершин. Всякий циклический маршрут нечетной длины содержит простой цикл нечетной длины. Всякий цикл содержит простой цикл.

СЛЕДСТВИЕ:

Всякий кратчайший маршрут между двумя заданными вершинами графа есть простая цепь. Всякий цикл наименьшей длины при заданной вершине является простым.

1.6 Определение числа маршрутов длины «L» на графе

Маршрутом $\mu_{i,j}$ в графе G=(X,U) называется конечная последовательность вершин и рёбер вида –

$\mu_{0,l}$ =($x_0,u_1,x_1,u_2,x_2,...,x_{l-1},u_k, x_l$),

где x_0, x_l – соответственно начальная и конечная вершины маршрута $\mu_{0,l}$.

Очевидно, в конечном графе G=(X,U) можно выделить только конечное число маршрутов. Длина маршрута $\mu_{i,j}$ равна числу рёбер, которые в него входят.

Часто требуется знать, сколько маршрутов заданной длины в графе G связывает вершину x_i с вершиной x_j .

Для определения маршрутов длины q в графе G=(X,U) его матрицу смежности R возводят в степень, равную q. Тогда для каждого значения степени q=1,2,…,k значение элемента $(r_{i,j})^q$ матрицы R^q определяет количество маршрутов $\mu_{i,j}$ длиной, равной значению степени q.

<u>*ПРИМЕР.*</u> Для графа G= (X,U) , представленного на рисунке 3, определить количество маршрутов длины, равной 2.

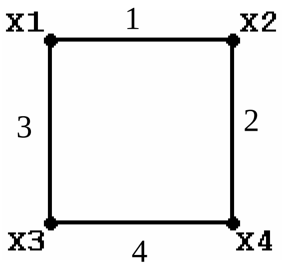


Рисунок 3

Матрица смежности R графа G имеет вид:

R=

	X1	X2	X3	X4
X1	0	1	1	0
X2	1	0	0	1
X3	1	0	0	1
X4	0	1	1	0

Возведем матрицу R в квадрат:

R^2=

	X1	X2	X3	X4
X1	2	0	0	2
X2	0	2	2	0
X3	0	2	2	0
X4	2	0	0	2

Значение каждого элемента $r_{i,j}$ матрицы R^2 равно числу маршрутов длины 2, ведущих из вершины x_i в вершину x_j.

Например, $r_{3,2}$=2 означает, что в графе два маршрута длины 2, которые ведут из вершины x_3 в вершину x_2 . Запишем их:

$\mu_{3,2}=x_3,3,x_1,1,x_2$; $\mu_{3,2}=x_3,4,x_4,2,x_2$.

Выполнить индивидуальное задание:

1. Построить матрицы смежности и инциденций для графа G=(X,U) своего варианта (рисунок 1).
2. По матрицам, представленным на рисунках 2, 3, построить графы, предварительно определив их тип в терминах теории графов. Определить класс построенных графов.
3. Определить число маршрутов длины L = 3 для графа G =(X,U) своего варианта.
4. Построить все маршруты длины L = 3 между вершинами, указанными преподавателем (помечены символом *).
5. Произвести ориентацию (произвольно) графа своего варианта.

Тема 2. УНАРНЫЕ ОПЕРАЦИИ НА ГРАФЕ

2.1 Удаление вершины

При удалении вершины удаляются все инцидентные ей рёбра.

Пример.

В графе G = (X,U) на рисунке 1 удалить вершину x1.

Граф G((рисунок 2) – результат выполнения данной операции.

 EMBED Word.Picture.8

Рисунок 1 Рисунок 2

2.2 Удаление ребра

При удалении ребра инцидентные ему вершины (концевые) не удаляются!

Пример.

В графе G = (X,U) на рисунке 1 удалить ребро u2.

Граф G((рисунок 3) – результат выполнения данной операции.

 EMBED Word.Picture.8

Рисунок 3

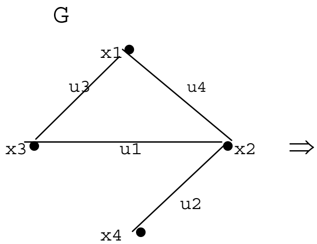
Если из графа требуется удалить некоторое множество вершин и рёбер, то эта процедура сводится к последовательному удалению каждой вершины отдельно и отдельно каждого ребра.

2.3 Замыкание (отождествление) вершин

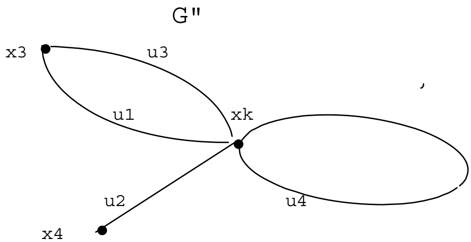
Для любой заданной пары вершин V_i, V_j операция замыкания сводится к отождествлению этих вершин в новую вершину V_k , при этом все рёбра, инцидентные вершинам V_i и V_j , становятся инцидентными вершине V_k .

ПРИМЕР. На графе G=(X,U), представленном на рисунке 4, а, выполнить операцию «замыкания» вершин x_1 и x_2.

На рисунке 4, б представлен граф G″, полученный из графа G после «замыкания» вершин x1 и x2, где вершина xk=(x1+x2).



а)



б)

Рисунок 4

2.4 Стягивание вершин графа по ребру

Операция стягивания вершин x_i и x_j графа G(X,U) по инцидентному им ребру u_k включает операцию удаления ребра u_k и операцию отождествления вершин x_i, x_j .

ПРИМЕР. На рисунке 5, б представлен граф G′, полученный из графа G (рисунок 5, а) операцией стягивания вершин x3 – x2 по ребру u1.

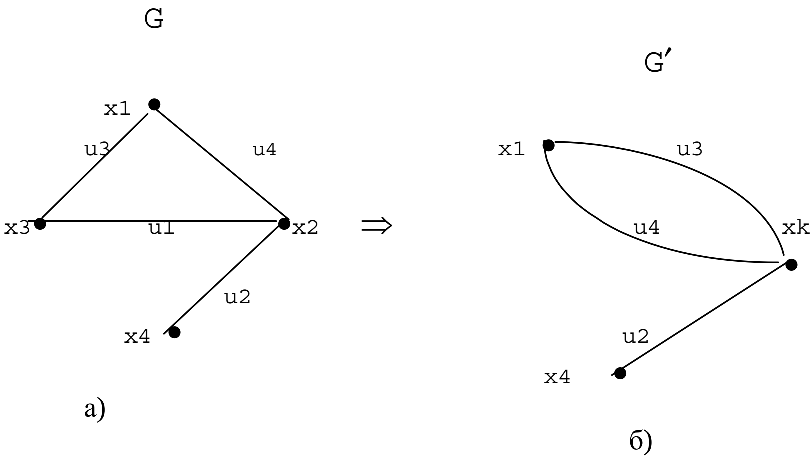


Рисунок 5

ЗАДАНИЕ

Выполнить унарные операции на графе своего варианта.

Результат выполнения операций показать на матрицах смежности или инциденций.

<u>Тема 3.</u> ЧАСТИ ГРАФА

3.1 Подграф

Граф G′ = (X′, U′) является подграфом графа G = (X,U), если X′ является подмножеством X и U′ является подмножеством U.

<u>Пример</u>.

1. Граф G1 (рисунок 11) является подграфом графа G (рисунок 10).
2. Граф G2 (рисунок 12) является подграфом графа G (рисунок 10).

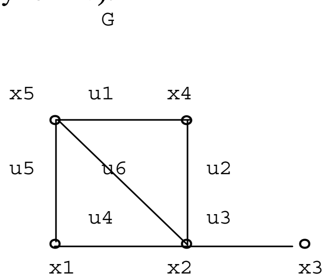


Рисунок 10

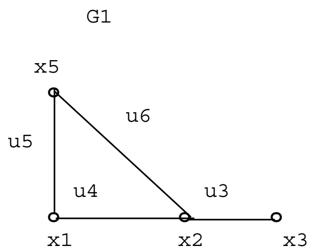


Рисунок 11

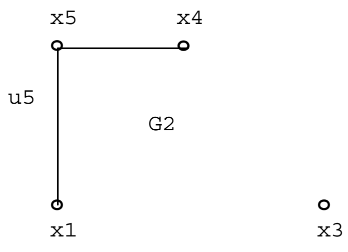


Рисунок 12

3.2 Суграф (частичный граф)

Граф $L' = (X', Y')$ является суграфом графа $L(X, U)$, если:

1. $X' = X$;
2. U' является подмножеством U, т.е. $U' \subset U$.

<u>Примеры</u>.

1. Граф G2 (рисунок 13) является суграфом (частичным графом) графа G (рисунок 10).
2. Граф L1 (рисунок 15) является суграфом (частичным графом) графа L = (рисунок 14).

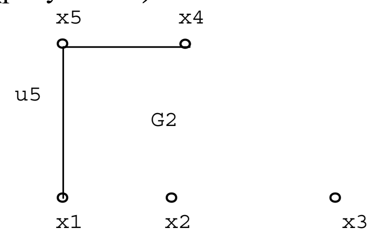


Рисунок 13

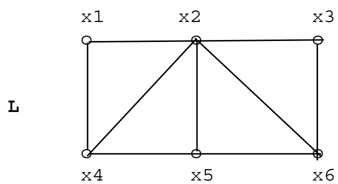


Рисунок 14

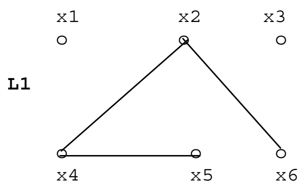


Рисунок 15

ЗАДАНИЕ:

1. Построить одновершинные, двувершинные, тривершинные подграфы для графа, данного в индивидуальном задании на рисунке 1.
2. Построить суграфы (частичные графы) для графа, данного в индивидуальном задании на рисунке 1.

Тема 4. ВЗВЕШЕННЫЕ ГРАФЫ. МЕТРИКА ГРАФА

Задавая на вершинах и рёбрах графа L=(X,U) функции p: $X \to M_p$, q: $U \to M_q$, где M_p и M_q – произвольные множества, получим взвешенный граф G(p,q)= (X,U,p,q).

На множествах X и U можно задавать и более чем по одной функции или, напротив, задать функцию только на рёбрах.

К взвешенным графам принадлежат электрические схемы, сети коммуникаций, информационные и логические сети, графы автоматов, сетевые графики работ и многое другое. Ограничимся здесь отдельным вопросом, в котором наличие весов является идеей чистой теории графов: длины рёбер. Пусть L(q) = (X,U; q) – обыкновенный граф с весовой функцией q, относящей каждому ребру $u\in U$ действительное число q(u)>0 в качестве длины. Если M – маршрут на графе L, то сумма q(M)= $\sum_{u\in M} q(u)$ по всем его рёбрам называется его q-длиной, а просто «длина» понимается как количество рёбер маршрута (каждое ребро графа надо считать столько раз, сколько оно встречается в маршруте). Число

$$\rho(x,y,)=\min\{q(M)/M\in M(x,y)\} \ (*),$$

где M(x,y) – множество всех простых цепей из x в y, называется *расстоянием* между вершинами $x,y\in X$ взвешенного графа L(q); если x = y, то M – цепь нулевой длины и её длина q(M)=0, а если вершины x и y отделены в графе, то $\rho(x,y,)= +\infty$.

Термин «расстояние» оправдан тем, что функция ρ, определённая посредством выражения (*), удовлетворяет трём аксиомам Фрише:

$$\forall x,y\in X[\rho(x,y)=0\Rightarrow x=y], \quad (1)$$

$$\forall x,y\in X[\rho(x,y,)= \rho(y,x)], \quad (2)$$

$$\forall x,y\in X[\rho(x,y)+ \rho(y,z)= \rho(x,z)], \quad (3)$$

т.е. ρ является метрикой на множестве вершин X. В частном случае, когда все q(u)=1 и, значит, q-длина всякой цепи совпадает с её обычной длиной, метрика $\rho = \rho_L^1$ графа L[1] называется *естественной метрикой* обыкновенного графа L=(X,U).

Вершина $x_0 \in X$ графа $L=[q]$ называется *центральной*, если

$$\forall x \in X[\max_{y \in X} \rho(x,y) \geq \max_{y \in X} \rho(x_0,y)];$$

Вершина $x_0 \in X$ графа $G=[q]$ называется *периферийной*, если

$$\forall x \in X[\max_{y \in X} \rho(x,y) \leq \max_{y \in X} \rho(x_0,y)].$$

В силу того, что множество X конечно, а величина $+\infty$ допускается как возможное значение функции ρ, вершины каждого из двух указанных типов всегда существуют. Величина

$$r(G)=\min_{x \in X} \max_{y \in X} \rho(x,y)$$

носит название *радиуса*, а величина

$$d(G)= \max_{x,y \in X} \rho(x,y)$$

называется *диаметром* графа $L(X,U)$. У несвязного графа $\max \rho(x,y)=+\infty$ для любой пары вершин $x,y \in X$, поэтому каждая его вершина x является одновременно и центральной, и периферийной, а радиус и диаметр бесконечны.

ПРИМЕР. Дан граф $L=(X,U)$ (рисунок 1) с естественной метрикой ρ.

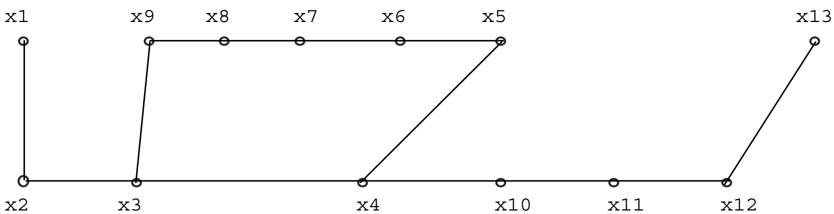


Рисунок 1

У данного графа вершины x4 и x10 – центральные, вершины x1, x7, x8, x13 – периферийные, $r(L)=4$, $d(L)=7$.

Способ нахождения метрики графа

Для нахождения метрики $\rho = \rho_L^1$ графа L = (X,U) достаточно знать его матрицу смежности R={ $r_{i,j}$} над булевой алгеброй B = (0,1), т.е. элементы матрицы $r_{i,j}$ = 1, если вершины x_i и x_j – смежны и $r_{i,j}$ = 0, в противном случае, все действия над элементами матрицы R производятся по правилам *логической алгебры:*

$$1 + 1 = 1;\ 0 + 0 = 0;\ 1 + 0 = 1;\ 0 * 0 = 0;\ 1 * 0 = 0.$$

Сопоставляя уже известные нам способы для установления существования маршрутов в графе длины q = m, можно утверждать, что при *возведении в степень* матрицы S = R + E, где E – единичная матрица той же размерности, что и размерность матрицы R, *на некотором шаге возведения в степень получим*:

$S = S^{k+1}$, т.е. *устойчивую матрицу* S в степени «k».

Значения степеней p матрицы S^p: p= {k, k–1, k–2, ... , 1} *равны длинам простых кратчайших цепей*, связывающих вершины x_i и x_j.

Таким образом, последовательно возводя в степень p = {1, 2, 3,…, k} матрицу S до получения устойчивой матрицы S^k, можно определить расстояния между всеми вершинами графа L=(X,U), построив матрицу метрики графа L.

Алгоритм построения матрицы метрики графа

Исходные данные для построения матрицы метрики (отклонений):

1. Граф L=(X,U).
2. Матрица смежности R графа L с элементами логического типа:

$$r_{i,j} = \begin{cases} 1, \text{ если вершины } x_i, x_j \text{ – смежны;} \\ 0 \text{ в противном случае.} \end{cases}$$

Введем обозначения:

R – матрица смежности заданного графа L;

E – единичная матрица;

M – матрица метрики (отклонений).

Описание алгоритма

Матрица метрики M= ($m_{i,j}$) строится за несколько итераций по результатам последовательного возведения матрицы R=(E+R) в степень q=$\overline{1,k}$ до получения устойчивой матрицы R^k, где k – степень устойчивой матрицы R^k. Матрица R^k называется устойчивой, если $R^k = R^{k+1}$.

Размерность матрицы M равна размерности матрицы R.

Все элементы матрицы M не определены.

Шаг 1.

Степень q матрицы R равна «1»: q=1.

$\forall\ m_{i,i}$ присваиваем значение «0», на основании аксиомы Фрише.

Шаг 2. Всем элементам $m_{i,j}$, значения которых не определены, присвоить значение степени q, если соответствующие им элементы матрицы $R^q \neq 0$. (*Не забывайте, что значения элементов* $m_{i,j}$ *определяются только один раз*).

Шаг 3. Проверяем, имеются ли в матрице M элементы $m_{i,j}$, значения которых ещё не определены?

Если такие элементы имеются, то переходим к шагу 4; в противном случае – к шагу 7.

Шаг 4. Повышаем степень q матрицы R: q=q+1.

Шаг 5. Проверяем, является ли матрица R^{q-1} устойчивой.

Если матрица R^{q-1} – неустойчивая, то переходим к шагу 2.

Иначе – переходим к шагу 6.

Шаг 6. Всем элементам $m_{i,j}$ матрицы M, **значения которых остались неопределенными**, присваиваем значение ∞ (бесконечность). Это говорит о том, что граф L=(X,U) несвязный.

Шаг 7. Матрица метрики M=($m_{i,j}$) построена. Конец алгоритма.

Примечание: 1. **Элементам $m_{i,j}$ значения присваиваются только один раз! Следовательно, если элемент $m_{i,j}$ уже определён, то его значение не меняется****.**

2. ***Радиус*** графа определяется по матрице метрики следующим способом: **в каждой строке** матрицы М выделяется максимальный элемент. Минимальный элемент из этих максимальных и есть радиус графа.

3. ***Диаметр*** графа определяется по матрице метрики следующим способом: **в каждой строке** матрицы М выделяется максимальный элемент. Максимальный элемент из этих максимальных и есть диаметр графа.

ЗАДАНИЕ

1. Построить метрику графа для своего варианта индивидуального задания.
2. Вычислить радиус, диаметр данного графа.
3. Найти все периферийные и центральные вершины графа.

Тема 5. СТРУКТУРНЫЙ АНАЛИЗ ГРАФОВ

Задача структурного анализа графов является одной из центральных задач теории графов и имеет широкое применение при решении фундаментальных теоретических проблем в программировании и прикладных задачах при анализе объектов математических моделей.

Данная задача связана с базовыми понятиями: связность графа; компонента связности графа, число компонент связности, полный (плотный) граф, максимальные полные и максимальные пустые подграфы, скелет графа, определения которых и теоретические выкладки даны в учебном пособии «Дискретная математика. Часть 2» (автор Е.Ф. Жигалова), а также в темах 1, 3 данных методических указаний.

В данной теме мы более подробно рассмотрим алгоритмы для нахождения всех максимальных пустых и максимальных полных подграфов в заданном графе общего вида.

Определение: а) *подграф называется максимальным пустым подграфом графа L=(X,U;P), если он не является подграфом никакого большего максимального пустого подграфа заданного графа;*

б) *подграф называется максимальным полным подграфом графа L=(X,U;P), если он не является подграфом никакого большего максимального полного подграфа заданного графа.*

Легко доказать, что задача выявления максимальных полных (плотных) и максимальных пустых подграфов в заданном графе $L=(X,U;P)$ общего вида легко сводится к случаю обыкновенных графов. Поэтому, для практического выявления всех максимальных полных и пустых подграфов в произвольном графе, достаточно уметь выявлять только максимальные плотные и пустые подграфы обыкновенных графов.

Приведём алгоритм выявления всех максимальных пустых подграфов в заданном графе общего вида, основанный на работах Х. Магу и Дж. Уэйсмана:

п.1.Для графа L=(X,U;P) общего вида построим его скелет $\overline{L}=(X,\overline{U})$ (смотри тему 1 данного методического пособия).

п.2. Построим матрицу инциденций А графа $\overline{L}=(X,\overline{U})$, элементы которой $a_{i,j}$ принимают значения 0 либо 1 ($i=\overline{1,n}$; $j=\overline{1,m}$, *где* n *=* $/X/$ *– число вершин в* $\overline{L}=(X,\overline{U})$, m *=* $/\overline{U}$ $/$*– число рёбер в* $\overline{L}=(X,\overline{U})$).

п.3. Дополним систему логическими переменными x_1, x_2, …, x_i, …, x_n , которые принимают значения 0 и 1, и подчиним её условиям:

$$x_i^2=x_i;\ x_i+1=1;\ x_i+x_i=x_i,\ 1+1=1,\ \text{т.е. } 2=1,\ (i=1,2,\ldots,n),$$

а также законам коммутативности, ассоциативности и дистрибутивности.

п.4. Из матрицы инциденций $A=\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ a_{21} & a_{22} & \ldots & a_{2m} \\ . & . & . & . \\ a_{n1} & a_{n2} & \ldots & a_{nm} \end{pmatrix}$

графа $\overline{L}=(X,\overline{U})$, где n ,m соответственно равны числу вершин и рёбер графа, образуем матрицу

$$Ax=\begin{pmatrix} a_{11}x_1 & a_{12}x_1 & \ldots & a_{12}x_1 \\ a_{21}x_2 & a_{22}x_2 & \ldots & a_{2m}x_2 \\ . & . & . & . \\ a_{n1}x_n & a_{n2}x_n & \ldots & a_{nm}x_n \end{pmatrix}$$

и составим произведение

$$\Pi_L=\Pi_L(x_1,x_2,\ldots,x_n)=\prod_{j=1}^{m}\sum_{i=1}^{n}a_{ij}x_i$$

Очевидно, что j-й сомножитель произведения Π_L есть сумма двух слагаемых, соответствующих тем двум вершинам, которые в графе соединены j-м ребром.

п.5. Преобразуем произведение Π_L к бесскобочному виду и привести всю сумму к минимальной форме, пользуясь дистрибутивным, ассоциативным, коммутативным законами и применяя закон поглощения: а) а+ав =а; б) (а+в)(а+с)$\cdots$(а+р) =

=а+вс¨¨р, где а,в,с,…,р – логические переменные, принимающие значения 0;1, выполняя при этом условия, описанные в п.3.

В результате выполненных преобразований выражение Π_L будет иметь минимальную форму и представлять сумму произведений переменных из множества x_1, x_2, …,x_i,…,x_n , т.е. многочлен. Обозначим его $\sum_L$.

п.6. Для каждого слагаемого многочлена $\sum_L$ выделим переменные, которые в него не входят, но входят в множество {x_1, x_2, …, x_i, …, x_n }. Эти переменные порождают максимальные пустые подграфы данного графа L, так как соответствующие им вершины графа L образуют максимальные пустые подграфы.

ПРИМЕР. В графе $L=(X,U;P)$, представленном на рисунке 1, выделить все максимальные пустые подграфы.

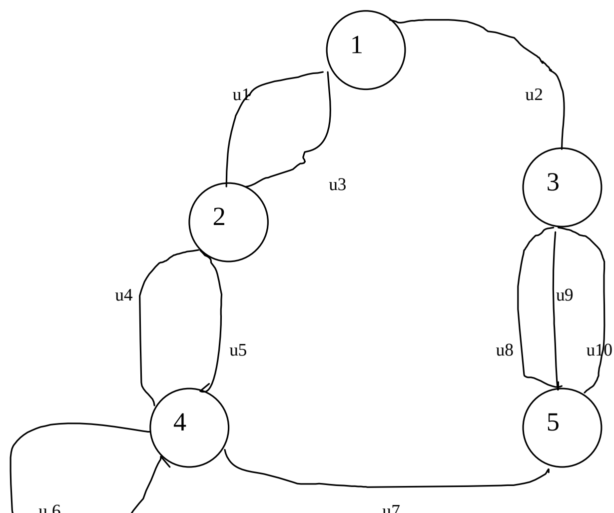


Рисунок 1

Матрица смежности B графа L содержит элементы (b_{ij}), $i=\overline{1,5}$; $j=\overline{1,5}$, равные:

$b_{11}= b_{22}= b_{33}= b_{55}=0$; $b_{44}=1$; $b_{12}=2$; $b_{13}=1$; $b_{14}=0$; $b_{15}=0$;
$b_{21}=2$; $b_{23}=0$; $b_{24}=2$; $b_{25}=0$; $b_{31}=1$; $b_{32}=0$; $b_{34}=0$; $b_{35}=3$;
$b_{41}=0$; $b_{42}=2$; $b_{43}=0$; $b_{45}=1$; $b_{51}=0$; $b_{52}=0$; $b_{53}=3$; $b_{54}=1$;

ш.1. Строим скелет $\overline{L}=(X,\overline{U})$ (рисунок 2) графа L.

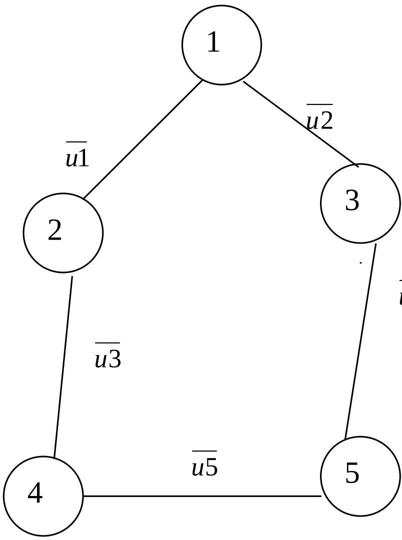


Рисунок 2

ш.2. Для графа $\overline{L}$ определим его матрицу инциденций А:

$$A = \begin{array}{c|ccccc} & \overline{u1} & \overline{u2} & \overline{u3} & \overline{u4} & \overline{u5} \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 4 & 0 & 0 & 1 & 0 & 1 \\ 5 & 0 & 0 & 0 & 1 & 1 \end{array}$$

ш.3. Введём новые логические переменные x_1, x_2, x_3, x_4, x_5 (по числу вершин в графе $\overline{L}$) и из матрицы A образуем матрицу Ax**:**

$$Ax = \begin{array}{c|ccccc} & \overline{u1} & \overline{u2} & \overline{u3} & \overline{u4} & \overline{u5} \\ \hline 1 & x1 & x1 & 0 & 0 & 0 \\ 2 & x_2 & 0 & x_2 & 0 & 0 \\ 3 & 0 & x_3 & 0 & x_3 & 0 \\ 4 & 0 & 0 & x_4 & 0 & x_4 \\ 5 & 0 & 0 & 0 & x_5 & x_5 \end{array}$$

ш.4. Составляем произведение $П_L$

$$П_L = (x_1 + x_2)(x_1 + x_3)(x_2 + x_4)(x_3 + x_5)(x_4 + x_5)$$

ш.5. Преобразуем выражения $П_L$ к минимальной форме:

$$П_L = (x_1 + x_2)(x_1 + x_3)(x_2 + x_4)(x_3 + x_5)(x_4 + x_5) =$$

(перемножаем скобки первую со второй и третью с пятой)

$$= (x_1 + x_2 x_3)(x_4 + x_2 x_5)(x_3 + x_5) =$$

(перемножаем скобки первую со второй)

$$= (x_1 x_4 + x_1 x_2 x_5 + x_2 x_3 x_4 + x_2 x_3 x_5)(x_3 + x_5) =$$

(перемножаем скобки первую со второй)

$$= x_1 x_3 x_4 + x_1 x_4 x_5 + x_1 x_2 x_3 x_5 +$$

$$+ x_1 x_2 x_5 + x_2 x_3 x_4 + x_2 x_3 x_4 x_5 + x_2 x_3 x_5 + x_2 x_3 x_5 =$$

(применяем законы, указанные в п.п. 3,5 данного пособия)

$= x_1x_3x_4 + x_1x_4x_5 + x_1x_2x_5 + x_2x_3x_4 + x_2x_3x_5$.

Преобразование выражения Π_L закончено. Получена минимальная форма-полином $\sum_L$.

ш.6. Выделим для каждого слагаемого полинома

$$\sum_L = x_1x_3x_4 + x_1x_4x_5 + x_1x_2x_5 + x_2x_3x_4 + x_2x_3x_5$$

его дополнение до множества переменных $\{x_1, x_2, x_3, x_4, x_5\}$:

(x_2x_5); (x_2,x_3); (x_3,x_4); (x_1,x_5); (x_1,x_4);

полученные дополнения порождают максимальные пустые подграфы графа $\overline{L}$ и заданного графа L.

Алгоритм Х. Магу и Дж. Уэйсмана может быть применён и для выявления в графе L=(X,U; P) общего вида всех максимальных полных (плотных) подграфов. Для этого необходимо построить для заданного графа $L = (X,U;P)$ его скелет – граф $\overline{L} = (X,\overline{U})$, а для графа $\overline{L} = (X,\overline{U})$ построить дополнительный граф $L^* = (X,U^*)$ (определение дополнительного графа дано в теме 1 данного методического пособия). Получить дополнительный граф легко, если исходный граф задать матрицей смежности его вершин, в которой всем элементам, равным нулю, присвоить значение «1», а всем элементам, значения которых не равны нулю, присвоить значение «0».

Далее для полученного графа $L^* = (X,U^*)$ с помощью алгоритма Х. Магу и Дж. Уэйсмана (рассмотренного выше) выявляем все максимальные пустые подграфы. Эти подграфы являются максимальными полными (плотными) подграфами для графов $\overline{L} = (X,\overline{U})$ и $L = (X,U;P)$.

ПРИМЕР. В графе $L = (X,U;P)$, представленном на рисунке 1, выделить все максимальные полные(плотные) подграфы.

ш.1. Строим скелет $\overline{L} = (X,\overline{U})$ (рисунок 2) графа L.

ш.2. Для графа $\overline{L}=(X,\overline{U})$ строим его дополнительный граф $L^{*}=(X,U^{*})$ (рисунок 3), в котором с помощью алгоритма Х.Магу–Дж.Уэйсмана выявляем максимальные пустые подграфы.

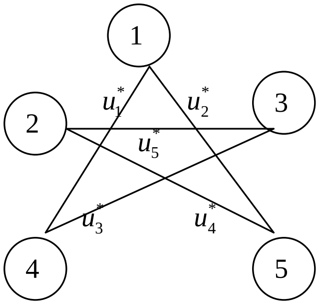


Рисунок 3

Приведём окончательный результат решения данной задачи – полином

$$\sum_L = x_1x_2x_3+x_1x_2x_4+x_1x_3x_5+x_2x_4x_5+x_3x_4x_5$$

и дополнения для его слагаемых: (x_4x_5); (x_3x_5); (x_2x_4); (x_1x_3); (x_1x_2), которые порождают все максимальные пустые подграфы графа $L^{*}=(X,U^{*})$ и максимальные полные (плотные) подграфы графа $\overline{L}=(X,\overline{U})$ и заданного графа $L=(X,U;P)$.

ЗАДАНИЕ

На графе своего варианта выделить все максимальные пустые и максимальные полные (плотные) подграфы, с помощью алгоритма Х.Магу–Дж.Уэйсмана, и привести их геометрическое представление.

КОНТРОЛЬНАЯ РАБОТА №2

Контрольная работа №2 содержит 3 задания.
ЗАДАНИЕ 1. Решить задачу коммивояжёра.

Исходные данные:
Значения элементов матрицы расстояний:

ВАРИАНТ 1

a(1,1)=∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=125	a(2.2)=∞	a(3.2)=23	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)=∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)=∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38

a(5.1)=22	a(5.4)=162
a(5.2)=63	a(5.5)=∞
a(5.3)=34	

ВАРИАНТ 2

a(1,1)= ∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)= ∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)=∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38

a(5.1)=22	a(5.4)=16
a(5.2)=63	a(5.5)=∞
a(5.3)=34	

ВАРИАНТ 3

a(1,1)=∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)=∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=124	a(3.3)=∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=98	a(4.4)=∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38
a(5.1)=222	a(5.4)=16	a(5.2)=63	a(5.5)=∝
a(5.3)=34			

ВАРИАНТ 4

a(1,1)= ∞	a(2.1)=53	a(3.1)=32	a(4.1)=211
a(1.2)=25	a(2.2)= ∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)= ∞
a(1.5)=46	a(2.5)=75	a(3.5)=124	a(4.5)=38
a(5.1)=22	a(5.4)=16	a(5.3)=34	a(5.2)=63
a(5.5)= ∞			

ВАРИАНТ 5

a(1,1)= ∞	a(2.1)=53	a(3.1)=32	a(4.1)=11	
a(1.2)=25	a(2.2)= ∞	a(3.2)=32	a(4.2)=35	
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29	
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)= ∞	
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38	
a(5.1)=22	a(5.4)=16	a(5.2)=63	a(5.5)= ∞	a(5.3)=234

ВАРИАНТ 6

a(1,1)= ∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)= ∞	a(3.2)=12	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)= ∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38
a(5.1)=22	a(5.4)=16		
a(5.2)= 3	a(5.5)= ∞	a(5.3)=34	

ВАРИАНТ 7

(1,1)= ∞	a(2.1)=153	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)= ∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)= ∞
a(1.5)=46	a(2.5)=75	a(3.5)=124	a(4.5)=38
a(5.1)=22	a(5.4)=16		
a(5.2)=63	a(5.5)= ∞		
a(5.3)=34			

ВАРИАНТ 8

a(1,1)=∞	a(2.1)=153	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)=∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)=∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)=∞
a(1.5)=46	a(2.5)=75	a(3.5)=124	a(4.5)=38

a(5.1)=22 a(5.4)=16 a(5.2)=63 a(5.5)=∞ a(5.3)=34

ВАРИАНТ 9

a(1,1)=∞ ; a(2.1)=53 ; a(3.1)=32; a(4.1)=11; a(1.2)=25; a(2.2)=∞ ; a(3.2)=72; a(4.2)=35; a(1.3)=115; a(2.3)=24; a(3.3)=∞ ; a(4.3)=129; a(1.4)=13; a(2.4)=36; a(3.4)=18; a(4.4)=∞ ; a(1.5)=46; a(2.5)=75; a(3.5)=24; a(4.5)=38; a(5.1)=22; a(5.4)=16; a(5.2)=63; a(5.5)=∞ ; a(5.3)=34

ВАРИАНТ 10

a(1,1)= ∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)= ∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)= ∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38
a(5.1)=22	a(5.4)=16	a(5.2)=63	a(5.5)= ∝
a(5.3)=34			

ВАРИАНТ 11

a(1,1)= ∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=95	a(2.2)=∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)= ∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38
a(5.1)=22	a(5.2)= 3	a(5.3)=34	a(5.4)=16
a(5.5)= ∞			

ВАРИАНТ 12

a(a(1,1)= ∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)= ∞	a(3.2)=72	a(4.2)=45
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)=∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)= 17
a(5.1)=22	a(5.4)=16	a(5.2)=43	a(5.5)= ∞
a(5.3)=34			

ВАРИАНТ 13

a(1,1)= ∞	a(2.1)=53	a(3.1)=32	a(4.1)=71
a(1.2)=25	a(2.2)= ∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)= ∞
a(1.5)=46	a(2.5)=25	a(3.5)=24	a(4.5)=38
a(5.1)=22	a(5.4)=16	a(5.2)=63	a(5.5)=∞
a(5.3)=34			

ВАРИАНТ_14

a(1,1)=∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)=∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=79
a(1.4)=13	a(2.4)=36	a(3.4)=118	a(4.4)=∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38
a(5.1)=22	a(5.4)=16		
a(5.2)=13	a(5.5)=∞		
a(5.3)=34			

ВАРИАНТ 15

(1,1)=∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=52	a(2.2)=∞	a(3.2)=32	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)=∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)=∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38

a(5.1)=22 a(5.4)=16 a(5.2)=36 a(5.5)=∞ a(5.3)=34

ВАРИАНТ 16

a(1,1)=∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)=∞	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)=∞	a(4.3)=29
a(1.4)=73	a(2.4)=36	a(3.4)=88	a(4.4)=∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38

a(5.1)=22	a(5.4)=16
a(5.2)=63	a(5.5)=∞
a(5.3)=34	

ВАРИАНТ 17

a(1,1)=∝	a(2.1)=53	a(3.1)=32	a(4.1)=81
a(1.2)=25	a(2.2)=∝	a(3.2)=72	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)=∝	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)=∝
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38
a(5.1)=22	a(5.4)=76	a(5.2)=63	a(5.5)=∝
a(5.3)=34			

ВАРИАНТ 18

a(1,1)=∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)=∞	a(3.2)=12	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)=∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)=∞
a(1.5)= 6	a(2.5)=57	a(3.5)=24	a(4.5)=38
a(5.1)=22	a(5.4)=16	a(5.3)=34	
a(5.2)=63	a(5.5)=∞		

ВАРИАНТ 19

a(1,1)=∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)=∞	a(3.2)=21	a(4.2)=35
a(1.3)=15	a(2.3)=24	a(3.3)= ∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)=∞
a(1.5)=46	a(2.5)=15	a(3.5)=24	a(4.5)=38

a(5.1)=22 a(5.4)=16 a(5.5)=∞ a(5.2)=63 a(5.3)=34

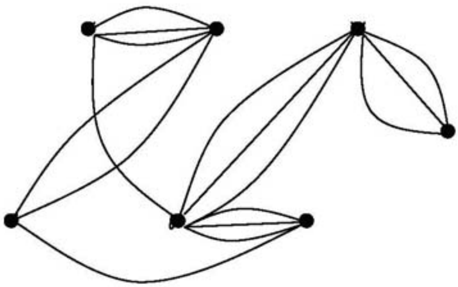
ВАРИАНТ 20

a(1,1)= ∞	a(2.1)=53	a(3.1)=32	a(4.1)=11
a(1.2)=25	a(2.2)=∞	a(3.2)=72	a(4.2)=95
a(1.3)=65	a(2.3)=24	a(3.3)=∞	a(4.3)=29
a(1.4)=13	a(2.4)=36	a(3.4)=18	a(4.4)=∞
a(1.5)=46	a(2.5)=75	a(3.5)=24	a(4.5)=38

a(5.1)=22 a(5.4)=16 a(5.2)=23 a(5.5)=∞ a(5.3)=34

ЗАДАНИЕ 2. Найти минимальную раскраску графа своего варианта с помощью алгоритма Магу. Определить хроматическое число.

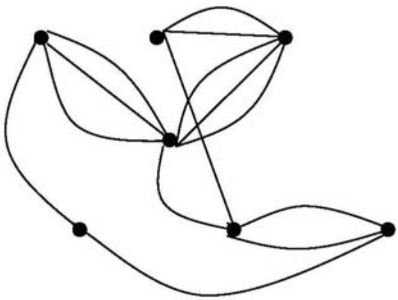
Вариант 1



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

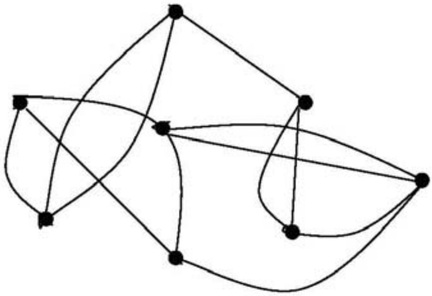
Вариант 2



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

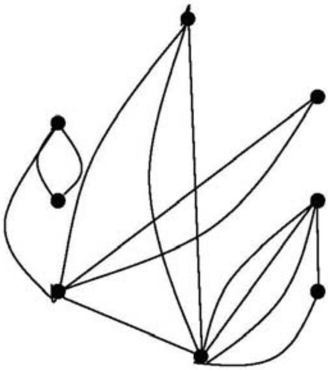
Вариант 3



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

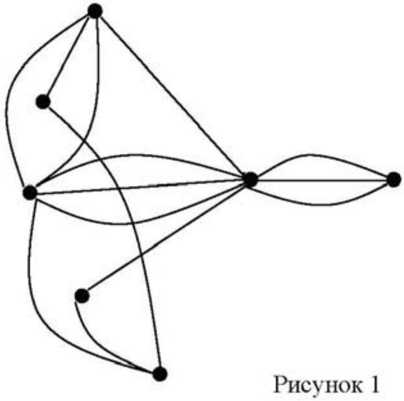
Вариант 4



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

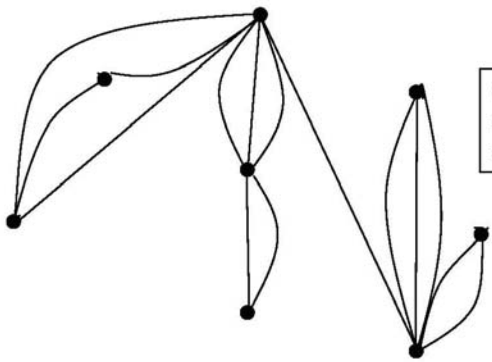
Вариант 5



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

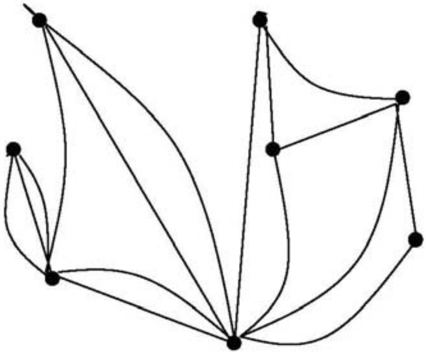
Вариант 6



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

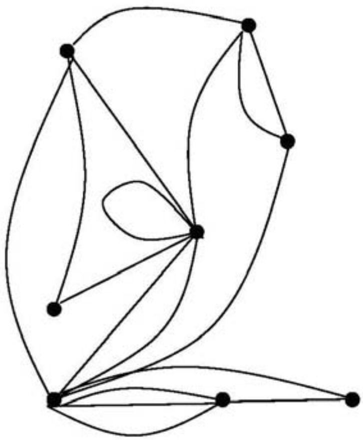
Вариант 7



Сделайте разметку вершин и ребер графа в произвольном порядке

Рисунок 1

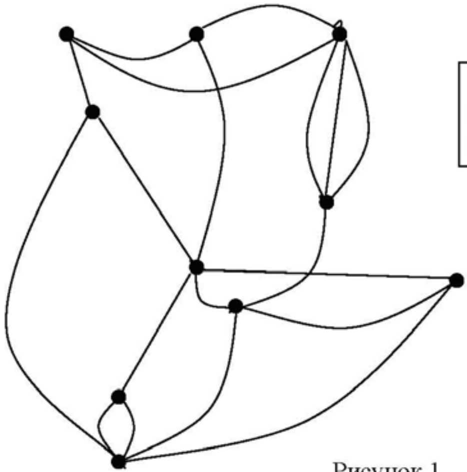
Вариант 8



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

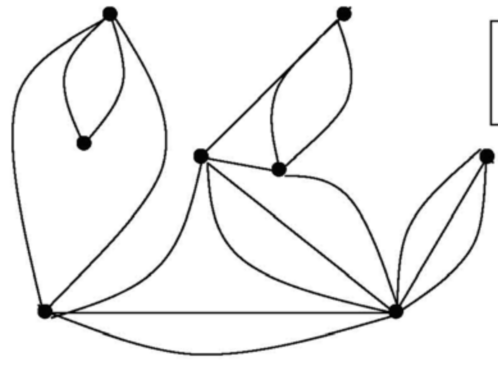
Вариант 9



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

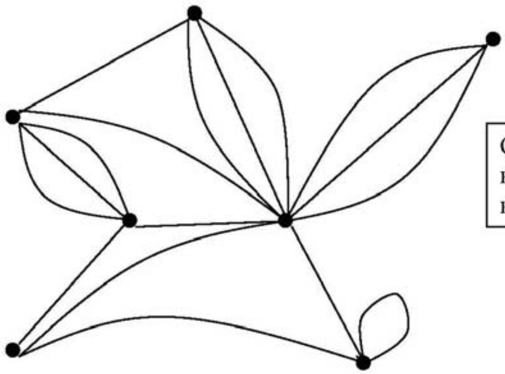
Вариант 10



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

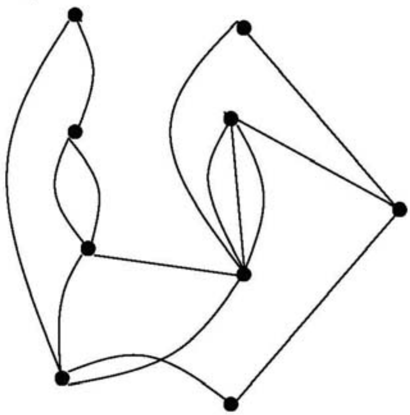
Вариант 11



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

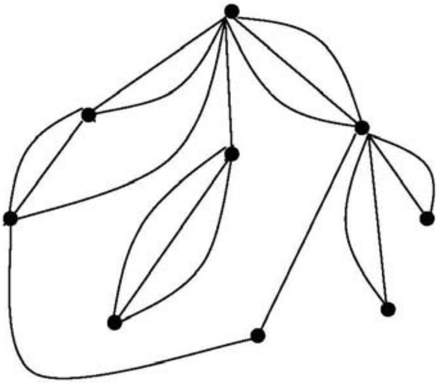
Вариант 12



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

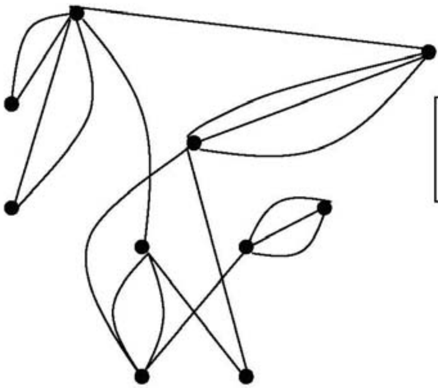
Вариант 13



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

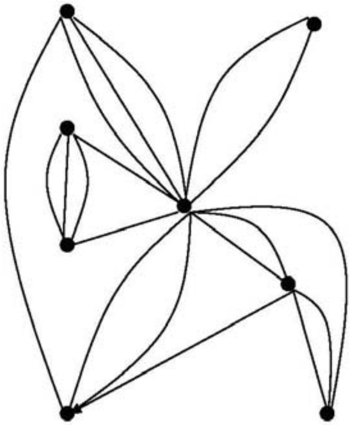
Вариант 14



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

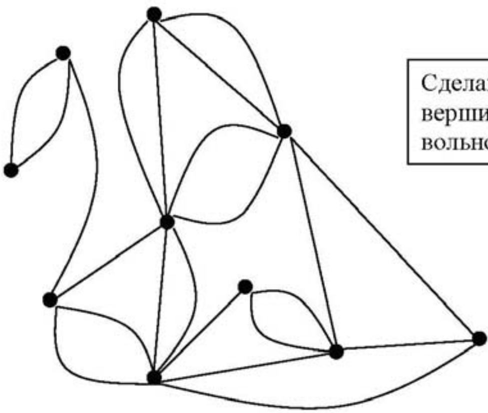
Вариант 15



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

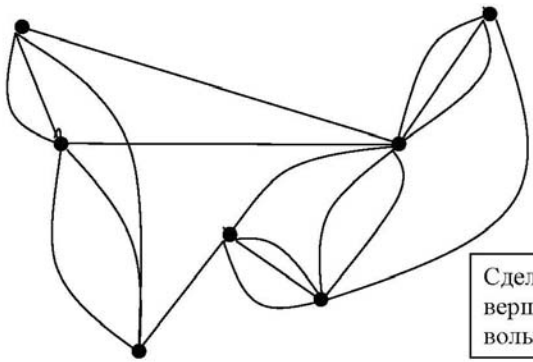
Вариант 16



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

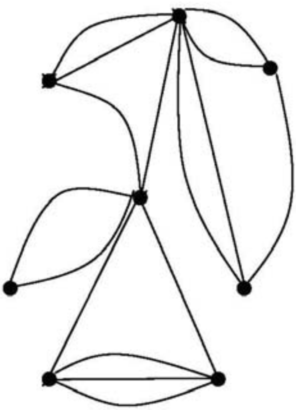
Вариант 17



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

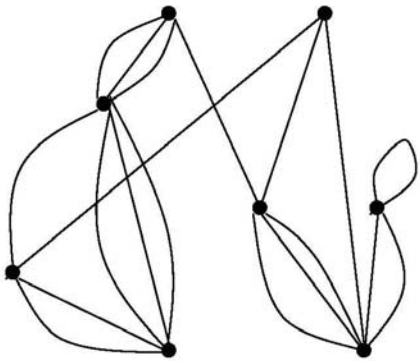
Вариант 18



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

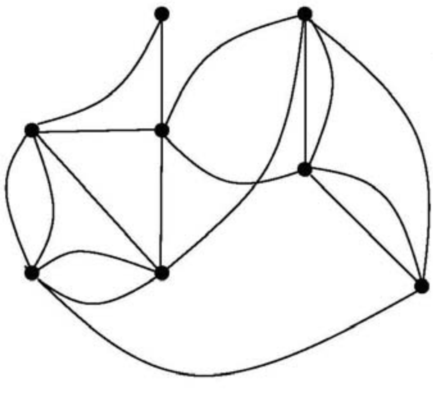
Вариант 19



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

Вариант 20



Сделайте разметку вершин графа в произвольном порядке

Рисунок 1

ЗАДАНИЕ 3.

ЗАДАЧА о максимальном потоке на сети.

Исходные данные:

Дана сеть S(X,U)

x_0 – исток сети; x_7 – сток сети, где $x_0 \in X$; $x_7 \in X$.

1. Вычислить значение максимального потока на сети S, с помощью алгоритма

Форда–Фалкерсона.

2. Построить минимальный разрез сети S.

Варианты значений пропускных $r_{i,j}$ способностей дуг сети (значения пропускных способностей дуг $r_{i,j}$ заданы по направлению ориентации дуг: от индекса i к индексу j):

ВАРИАНТ 1

r[0,1] = 19	r[4,7] = 14	r[6,3] = 13	r[5,7] = 43
r[0,2] = 10	r[4,2] = 18	r[6,7] = 35	r[5,4] = 26
r[0,3] = 9	r[2,5] = 21	r[2,1] = 61	r[6,5] = 41
r[1,4] = 15	r[2,6] = 35	r[3,2] = 32	

ВАРИАНТ 2

r[0,1] = 19	r[4,7] = 34	r[6,3] = 13	r[5,7] = 43
r[0,2] = 10	r[4,2] = 18	r[6,7] = 35	r[5,4] = 26
r[0,3] = 9	r[2,5] = 21	r[2,1] = 11	r[6,5] = 41
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 3

r[0,1] = 12	r[4,7] = 4	r[6,3] = 13	r[5,7] = 43
r[0,2] = 10	r[4,2] = 18	r[6,7] = 35	r[5,4] = 26
r[0,3] = 9	r[2,5] = 21	r[2,1] = 3	r[6,5] = 11
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 4

r[0,1] = 29	r[4,7] = 64	r[6,3] = 13	r[5,7] = 43
r[0,2] = 32	r[4,2] = 58	r[6,7] = 35	r[5,4] = 36
r[0,3] = 102	r[2,5] = 21	r[2,1] = 81	r[6,5] = 41
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 5

r[0,1] = 12	r[4,7] = 4	r[6,3] = 13	r[5,7] = 33
r[0,2] = 10	r[4,2] = 18	r[6,7] = 32	r[5,4] = 26
r[0,3] = 9	r[2,5] = 21	r[2,1] = 3	r[6,5] = 11
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 6

r[0,1] = 19	r[4,7] = 34	r[6,3] = 13	r[5,7] = 33
r[0,2] = 10	r[4,2] = 18	r[6,7] = 35	r[5,4] = 46
r[0,3] = 23	r[2,5] = 21	r[2,1] = 81	r[6,5] = 41
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 7

r[0,1] = 12	r[4,7] = 4	r[6,3] = 13	r[5,7] = 33
r[0,2] = 10	r[4,2] = 18	r[6,7] = 32	r[5,4] = 26
r[0,3] = 19	r[2,5] = 21	r[2,1] = 3	r[6,5] = 11
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 8

r[0,1] = 12	r[4,7] = 4	r[6,3] = 13	r[5,7] = 33
r[0,2] = 12	r[4,2] = 18	r[6,7] = 32	r[5,4] = 26
r[0,3] = 19	r[2,5] = 21	r[2,1] = 3	r[6,5] = 11
r[1,4] = 21	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 9

r[0,1] = 32	r[4,7] = 4	r[6,3] = 13	r[5,7] = 43
r[0,2] = 10	r[4,2] = 18	r[6,7] = 35	r[5,4] = 26
r[0,3] = 29	r[2,5] = 21	r[2,1] = 3	r[6,5] = 61
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 10

r[0,1] = 11	r[4,7] = 4	r[6,3] = 13	r[5,7] = 38
r[0,2] = 12	r[4,2] = 18	r[6,7] = 32	r[5,4] = 26
r[0,3] = 17	r[2,5] = 21	r[2,1] = 6	r[6,5] = 11
r[1,4] = 21	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 11

r[0,1] = 32	r[4,7] = 4	r[6,3] = 13	r[5,7] = 43
r[0,2] = 84	r[4,2] = 18	r[6,7] = 35	r[5,4] = 26
r[0,3] = 91	r[2,5] = 21	r[2,1] = 73	r[6,5] = 61
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 12

r[0,1] = 11	r[4,7] = 12	r[6,3] = 13	r[5,7] = 38
r[0,2] = 12	r[4,2] = 18	r[6,7] = 32	r[5,4] = 26
r[0,3] = 17	r[2,5] = 12	r[2,1] = 6	r[6,5] = 11
r[1,4] = 21	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 13

r[0,1] = 19	r[4,7] = 34	r[6,3] = 13	r[5,7] = 43
r[0,2] = 10	r[4,2] = 18	r[6,7] = 35	r[5,4] = 26
r[0,3] = 29	r[2,5] = 21	r[2,1] = 11	r[6,5] = 41
r[1,4] = 23	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 14

r[0,1] = 19	r[4,7] = 14	r[6,3] = 13	r[5,7] = 43
r[0,2] = 10	r[4,2] = 18	r[6,7] = 35	r[5,4] = 26
r[0,3] = 9	r[2,5] = 21	r[2,1] = 81	r[6,5] = 41
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 15

r[0,1] = 29	r[4,7] = 14	r[6,3] = 13	r[5,7] = 43
r[0,2] = 32	r[4,2] = 18	r[6,7] = 35	r[5,4] = 36
r[0,3] = 9	r[2,5] = 21	r[2,1] = 81	r[6,5] = 41
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 16

r[0,1] = 19	r[4,7] = 44	r[6,3] = 13	r[5,7] = 83
r[0,2] = 10	r[4,2] = 18	r[6,7] = 75	r[5,4] = 46
r[0,3] = 23	r[2,5] = 21	r[2,1] = 81	r[6,5] = 41
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 17

r[0,1] = 19	r[4,7] = 34	r[6,3] = 13	r[5,7] = 23
r[0,2] = 15	r[4,2] = 18	r[6,7] = 35	r[5,4] = 26
r[0,3] = 20	r[2,5] = 21	r[2,1] = 11	r[6,5] = 41
r[1,4]=23	r[2,6]=15	r[3,2] = 32	

ВАРИАНТ 18

r[0,1] = 19	r[4,7] = 44	r[6,3] = 33	r[5,7] = 53
r[0,2] = 10	r[4,2] = 18	r[6,7] = 95	r[5,4] = 46
r[0,3] = 23	r[2,5] = 21	r[2,1] = 81	r[6,5] = 71
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 19

r[0,1] = 19	r[4,7] = 34	r[6,3] = 13	r[5,7] = 33
r[0,2] = 15	r[4,2] = 18	r[6,7] = 35	r[5,4] = 36
r[0,3] = 20	r[2,5] = 21	r[2,1] = 11	r[6,5] = 41
r[1,4] = 23	r[2,6] = 15	r[3,2] = 32	

ВАРИАНТ 20

r[0,1] = 21	r[4,7] = 29	r[6,3] = 13	r[5,7] = 43
r[0,2] = 10	r[4,2] = 18	r[6,7] = 35	r[5,4] = 26
r[0,3] = 35	r[2,5] = 21	r[2,1] = 38	r[6,5] = 11
r[1,4] = 25	r[2,6] = 15	r[3,2] = 32	

<u>Тема 6.</u> МАРШРУТЫ СПЕЦИАЛЬНОГО ВИДА

К маршрутам специального вида относятся маршруты, относящиеся к понятию – обход графа. Обход графа это маршрут, содержащий все рёбра или все вершины данного графа и обладающий определёнными свойствами.

К обходам графа относятся эйлеровы цепи и циклы и гамильтоновы циклы.

Обходы графа вида эйлеровой цепи и эйлерова цикла достаточно подробно рассмотрены в учебном пособии «Дискретная математика. Часть 2» (автор Жигалова Е.Ф.) на страницах 56–59.

Остановимся на рассмотрении понятия обхода графа вида гамильтоновой цепи, цикла, пути и контура.

Гамильтоновой цепью в неориентированном графе называется цепь, проходящая через каждую его вершину один и только один раз.

Гамильтоновым циклом в неориентированном графе называется цикл, проходящий через каждую вершину один и только один раз за исключением начальной вершины, которая совпадает с конечной.

Гамильтоновым путём в ориентированном графе называется путь $S(x_0,x_1,\ldots,x_n)$, проходящий через все вершины графа, притом только по одному разу.

Гамильтоновым контуром называется контур $M(x_0,x_1,\ldots,x_n,x_o)$ в ориентированном графе $G=(X,U)$, если он проходит через все вершины графа, притом только по одному разу.

Существует несколько интерпретаций задачи о гамильтоновых циклах, имеющих ряд приложений в экономике и исследовании операций.

Рассмотрим одну из них – ***задачу о бродячем торговце*** (задачу о коммивояжёре). Район, который должен посетить коммивояжер, содержит определенное количество городов. Расстояния между ними известны, и нужно найти кратчайшую дорогу, проходящую через все пункты и возвращающуюся в исходный пункт. Решение данной задачи связано с нахождением гамильтонового цикла (контура) наименьшей длины в графе, интерпретирующем все возможные маршруты коммивояжёра.

Сформулирован целый ряд достаточных условий существования гамильтоновых цепей, циклов, путей и контуров. Приведем некоторые из них без доказательства.

1. Теорема Кёнига. В полном (плотном) конечном графе всегда существует гамильтонов путь.

2. Если в графе G(X,U) с n вершинами для любой пары вершин x_i, и $x_j \in X$ справедливо неравенство

$$m(x_i) + m(x_j) \geq n-1,$$

где – $m(x_i)$, $m(x_j)$ – степени вершин x_i, и x_j, то граф $G=(X,U)$ имеет гамильтонову цепь.

Несмотря на сходство в определении эйлерова и гамильтонова циклов, соответствующие теории для этих понятий имеют мало общего. Критерий существования для эйлеровых циклов установлен просто, для гамильтоновых циклов никакого общего правила неизвестно. Более того, иногда даже для конкретных графов бывает трудно решить, можно ли найти такой цикл. Дж. Литтлом для решения задачи коммивояжёра был предложен простой рекурентный алгоритм, основанный на идее метода ветвей и границ.

Рассмотрим его.

На содержательном уровне задача коммивояжера формулируется следующим образом. Для n городов, соединенных некоторыми путями, задана матрица C порядка n ($\dim C = n \times n$), для которой $c_{ij} \in C$ определяет стоимость (расстояние, время) пути из города i в город j. Требуется указать путь, включающий посещение всех городов, который имеет минимальное значение суммарной стоимости.

Произвольное решение представляет собой сумму n слагаемых, каждое из которых определяется элементом матрицы C в соответствии с принятым порядком посещения:

$$Z = c_{i_1 i_2} + c_{i_2 i_3} + c_{i_3 i_4} + \ldots + c_{i_{n-1} i_n} + c_{i_n i_1}$$

Оптимальным решением будет перестановка, минимизирующая эту сумму, т.е.

$$c_{i_1 i_2} + c_{i_2 i_3} + c_{i_3 i_4} + \ldots + c_{i_{n-1} i_n} + c_{i_n i_1} \to \min_{u \in U},$$

где $u = (i_1, i_2, \ldots, i_n)$ – перестановка для множества индексов $I = \{1, 2, \ldots, n\}$, $U = \{u\}$ – множество всех перестановок ($U = n!$).

В общем случае для некоторых пар i,j непосредственный переход от i к j может быть запрещен. Тогда элемент c_{ij} в матрице С полагается равным специальному значению – бесконечности: $c_{ij} = \infty$.

Если существует какое-то конечное решение задачи, то маршрут, соответствующий оптимальному решению, не содержит дуги с бесконечным значением. Поэтому в дальнейшем будем полагать, что некоторые элементы матрицы C могут быть равными ∞. Кроме того, положим, что все диагональные элементы матрицы C также равны ∞.

Дж. Литтлом был предложен простой рекуррентный алгоритм решения задачи коммивояжера. В основе этого алгоритма лежит последовательное преобразование составленных определенным образом матриц, приводящее каждый раз к одной из трех стандартных возможностей.

Сначала по данным решаемой задачи составляется исходная матрица. Затем эта матрица обрабатывается по определенной схеме, что приводит к более простым вариантам, также задаваемым в виде матриц. Повторное применение стандартных приемов к каждому из этих вариантов продолжается до тех пор, пока не будет получено окончательное решение задачи. Анализ каждой из матриц приводит к одной из трех возможностей:

1. Получению решения, когда решение находится непосредственно по исходной матрице, если задача достаточно проста.

2. Исключению матрицы из дальнейшего рассмотрения, когда можно показать, что из нее не следует решение задачи.

3. Ветвлению, состоящему в том, что решаемая задача приводится к рассмотрению двух вариантов более простых задач.

На каждом шаге описываемого алгоритма задача включает n городов, причем из n шагов маршрута – k шагов могут быть уже установлены, и нужно выбрать оптимальным образом оставшиеся

$n-k$ шагов. Для всех возможных маршрутов необходимо задать значение Y, представляющее собой нижнюю границу стоимости всех возможных решений задачи, включая оптимальное.

Особенность алгоритма Литтла состоит в использовании определенной эвристической схемы, позволяющей разумно построить эту нижнюю границу, стремясь сделать ее как можно больше.

Таким образом, задача характеризуется оставшимся числом $n-k$ неизвестных шагов маршрута и нижней границей Y стоимости решения задачи. Кроме того, можно считать, что известно, по крайней мере, одно решение исходной задачи. Обозначим через Z стоимость лучшего из уже найденных решений. Отметим, что при инициализации алгоритма Z может быть положено равным бесконечности, хотя более разумным подходом будет использование какого-либо приближенного алгоритма для (достаточно быстрого) определения некоторого приближенного значения Z.

Возможны случаи:

1. Если ***n–k = 2***, то осталось не более двух шагов маршрута, и решение находится сразу. Если его стоимость меньше ***Z***, то ***Z*** принимается равным этому новому значению, а решение считается лучшим из известных решений. Поэтому в литературе текущее значение ***Z*** называют ***рекордом***.

2. Если Y больше или равно ***Z***, то происходит ***отсечение*** и рассматриваемая задача исключается, поскольку представленные в ней *маршруты* не могут привести к решениям, лучшим, чем уже известное.

3. Если не имеет места ни одна из перечисленных выше ситуаций, то происходит *ветвление,* и вместо рассматриваемой задачи составляются две:

а) в первой из них определенным образом выбирается переход от i к j, в результате чего нижняя граница стоимости решений может возрасти;

б) во второй запрещается переход из i в j (элемент c_{ij} полагается равным ∞), в результате чего нижняя граница стоимости решений также может возрасти.

Таким образом, получаемые задачи характеризуются возрастающей нижней границей и (или) большим числом установленных шагов маршрута. Кроме того, для каждой последующей задачи число возможных маршрутов меньше, чем для предыдущей, и на некотором этапе достигается такое состояние, когда маршрут определен полностью.

Ситуации, когда решение получается сразу или задача исключается, очевидны. Особенность алгоритма заключается в ветвлении, суть которого составляют понятия ***приведения*** матрицы и ***выбора*** дуги.

Приведение матрицы C основано на том факте, что если из любой строки или любого столбца матрицы вычитается константа, то оптимальное решение не меняется, а стоимость оптимального решения отличается от исходного в точности на количество, вычтенное из строки или столбца. Поэтому если произведено вычитание, такое, что каждый столбец и строка содержат нуль и все c_{ij} неотрицательны, то общая вычтенная сумма будет нижней границей стоимости любого решения. Матрицу стоимостей, полученную после такого вычитания, будем называть ***приведенной,*** т.е. такая матрица не допускает дальнейшего приведения.

Пусть $\boldsymbol{C'}$ – матрица стоимостей задачи, рассматриваемой на текущем шаге алгоритма. Рассмотрим случай 3 (ветвление). После того как фиксирован переход из i в j, нужно модифицировать матрицу $\boldsymbol{C'}$. Обозначим матрицу новой задачи через $\boldsymbol{C' - Y}$. В этой матрице нужно исключить переходы из i во все другие города, кроме j, и переходы в j из всех других городов, кроме i. Для этого положим все элементы строки i и столбца j матрицы, за исключением c_{ij}, равными ∞.

Поскольку обход всех городов при единственном посещении каждого из них и окончания маршрута в исходной точке не может включать в себя одновременно дуги (i, j) и (j, i), необходимо положить $c_{ij} = \infty$. Кроме того, если ранее построенный частичный обход содержал пути

$\mathbf{i_1 \rightarrow i_2 \rightarrow \cdots \rightarrow i_u}$ **и** $\mathbf{j_1 \rightarrow j_2 \rightarrow \ldots \rightarrow j_v}$, **причём** $\mathbf{i = i_u}$ и $j = j_1$, то нужно запретить в последующем выбор дуги (j_v, i_1), полагая соответствующий элемент матрицы, равным ∞. Это позволит избежать повторного посещения городов до окончания обхода. Так как эти запрещения могут привести к устранению ряда нулей в матрице C', то не исключена возможность дальнейшего приведения матрицы $C' - Y$ и в результате этого получения новой большей границы для решений, связанных с этой матрицей.

Пусть переход из i в j запрещен. В этом случае также не исключена возможность дальнейшего приведения матрицы и вызванного этим возрастания нижней границы стоимости решений.

Выбор дуги (i, j) должен быть таким, чтобы максимально увеличить нижнюю границу и, возможно, исключить из рассмотрения ряд задач без дальнейшего ветвления. Чтобы достигнуть этого, просматриваются все возможные дуги (i,j) с нулевым весом и выбирается такая, чтобы сумма двух последовательных приводящих констант (по строке и по столбцу) была максимальной.

Алгоритм заканчивает работу, когда список задач, допускающих ветвление, исчерпан. Оптимальным является текущее рекордное решение.

Пример решения задачи коммивояжера алгоритмом Литтла даётся в программе GM', записанной на диске для специальностей 220300 (САПР) и 210100 (управление и информатика в технических системах).

Тема 7. РАСКРАСКА ГРАФОВ. ХРОМАТИЧЕСКОЕ ЧИСЛО

Теоретические положения по данной теме изложены в учебном пособии «Дискретная математика. Часть 2 » (Жигалова Е.Ф.).

Определение 1.

Рассмотрим граф $G = (X, U)$ и «раскрасим» его вершины так, чтобы смежные вершины не были окрашены в один цвет. Такую раскраску назовем *правильной.*

Если на правильную раскраску затрачено p цветов, то граф называется *p-хроматическим.*

Определение 2. Наименьшее натуральное число p, для которого граф является p-хроматическим, называется *хроматическим числом графа* и обозначается $\gamma(G)$.

Для решения задачи раскраски графа $G=(X,U)$ и определения его хроматического числа можно применить алгоритм Дж. Магу, который подробно изложен в теме 5 (структурный анализ графов) данного методического пособия.

Раскраска графа с помощью алгоритма Дж. Магу выполняется в два этапа.

На первом этапе определяются подмножества вершин графа G, которые можно раскрасить одним цветом.

На втором этапе определяется хроматическое число графа $\gamma(G)$.

Для определения подмножества вершин, которые можно раскрасить одним цветом, в графе $G=(X,U)$ с помощью алгоритма Дж. Магу находятся все максимальные пустые подграфы $G_1=(X_1,U_1)$, $G_2=(X_2,U_2)$, $G_3=(X_3,U_3)$, $\ldots,G_i=(X_i,U_i),\ldots$, $G_k=(X_k,U_k)$ (алгоритм подробно рассмотрен в теме 5 данного пособия). Очевидно, что вершины, принадлежащие одному подмножеству X_i, можно раскрашивать одним цветом. Наибольшее число цветов p, необходимое для раскраски вершин графа, равно числу k, т.е. числу максимальных пустых подграфов данного графа.

Так как подмножества вершин максимальных пустых подграфов могут пересекаться, то число р, как правило, больше хроматического числа $\gamma(G)$.

Для определения хроматического числа $\gamma(G)$ необходимо выполнить следующее:

1. Упорядочить все максимальные пустые подмножества X_1, $X_2, X_3, \dots, X_i, \dots, X_k$ графа G в порядке возрастания их кардинальных чисел $/X_i/$.
2. Выбрать X_i, имеющее $_{\max}/X_i/$.
3. Присвоить цвет (допустим, синий) всем вершинам $x_t \in {}_{\max}/X_i/$
4. Вычеркнуть из других подмножеств вершины, которым присвоен цвет.
5. Исключить из дальнейшего рассмотрения подмножество X_i.
6. Если оставшееся семейство пустых подмножеств X_1, X_2, $X_3, \dots, X_i, \dots, X_k$ пусто, то перейти к п.9, иначе – к п.7.
7. Из оставшихся подмножеств X_1, X_2, $X_3, \dots, X_i, \dots, X_k$ выбрать следующее $_{\max}X_s$, где $s \in \mathrm{I}{:}\mathrm{I} = \{1,2,3,\dots,k\}$, и присвоить вершинам, входящим в него, цвет, который еще не использовался.
8. Для $_{\max}X_s$ выполнить действия, описанные в п.п. 4, 5, 6, 7, принимая i=s.
9. Определить хроматическое число графа $\gamma(G)$.

$\gamma(G)$ = количеству цветов, потребовавшихся для раскраски в ходе выполнения всех действий, описанных в п.п. 1–8.

Пример. Найти раскраску для графа G=(X,U) на рисунке 7.1.

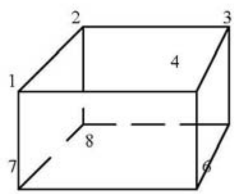


Рисунок 7.1

Решение.

1-й этап. Находим все максимальные пустые подграфы графа G=(X,U) с помощью алгоритма Дж. Магу.

1. Составляем произведение

$$P_G = \prod_{j=1}^{m} \sum_{i=1}^{n} a_{ij} \cdot x_i ,$$

где a_{ij} – элемент матрицы инциденций графа G; a_{ij}= (0,1); $x_i \in X$ – новые образующие, подчиняющиеся условиям $x_i^2 = x_i$, $x_i + 1 = 1$, 2=1 и законам коммутативности, ассоциативности и дистрибутивности, на основании которых преобразуем выражение к минимальной (бесскобочной) форме.

$$P_G = (x_1 + x_2)(x_1 + x_4)(x_1 + x_7)(x_3 + x_2)(x_3 + x_4)(x_3 + x_5)\times$$
$$\times(x_6 + x_4)(x_6 + x_5)(x_6 + x_7)\times(x_8 + x_2)(x_8 + x_5)(x_8 + x_7) =$$
$$= (x_1 + x_2x_4x_7)(x_3 + x_2x_4x_5)(x_6 + x_4x_5x_7)(x_8 + x_2x_5x_7) = \ldots =$$
$$= x_1x_3x_6x_8 + x_1x_2x_3x_5x_6x_7 + x_1x_3x_4x_5x_7x_8 + x_2x_4x_5x_7 +$$
$$+ x_1x_2x_4x_5x_6x_8 + x_2x_3x_4x_6x_7x_8.$$

2. Для каждого слагаемого преобразованного выражения P_G запишем его дополнение до полной системы образующих $\{x_i\}$, $i=\overline{1,8}$:

$$x_2x_4x_5x_7; \quad x_4x_8; \quad x_2x_6; \qquad x_1x_3x_6x_8; \quad x_3x_7; \quad x_1x_5.$$

Мы получили полный обзор всех максимальных пустых подграфов графа G.

Вершинам, принадлежащим одному максимальному пустому подграфу, приписываем один цвет. Таким образом, для правильной раскраски вершин графа G требуется шесть цветов, т.е. p=6.

Граф G является 6-хроматическим.

II-й этап решения задачи раскраски графа G – о*пределение хроматического числа* $\gamma(G)$ *графа G.*

Будем кодировать цвета арабскими символами: 1,2,3,4,5,6.

1.Упорядочим полученные множества вершин в порядке убывания их кардинальных чисел:

$x_2x_4x_5x_7$; $x_1x_3x_6x_8$; x_4x_8; x_2x_6; x_3x_7; x_1x_5.

2. Припишем вершинам множества $x_2x_4x_5x_7$ цвет «1» (цвет выбирается произвольно).

3. Удалим раскрашенные вершины из всех множеств: $x_1x_3x_6x_8$; x_8; x_6; x_3; x_1.

4. Припишем вершинам множества $x_1x_3x_6x_8$ цвет 2.

5. Удалим раскрашенные вершины из всех множеств – получаем пустое множество. Это говорит о том, что все вершины графа G – раскрашены. Для раскраски потребовалось всего два цвета, т.е. хроматическое число $\gamma(G)$ *графа G* равно двум.

Тема. 8 БИХРОМАТИЧЕСКИЕ ГРАФЫ

Обыкновенный граф L=(X, U) будем называть *графом Кёнига,* если множество ***X*** его вершин можно представить в виде объединения двух непересекающихся подмножеств $X^{'}, X^{''}$ так, чтобы никакие вершины одного и того же подмножества не были смежны, т.е.

$$X = X'\cup X'',\ X'\cap X'' = \varnothing,$$

$$\forall x,y\in X\left[x\widetilde{y}\in U \Rightarrow (x\in X^{'} \& y\in X^{''})\vee(x\in X^{''} \& y\in X^{'})\right].$$

Граф Кёнига часто записывают в виде *(X', X'';U);* такое задание не только указывает на возможность требуемого разбиения множества вершин, но и определяет конкретное разбиение; поэтому в случае, когда существуют по крайней мере два различных разбиения:

$$X = X_1^{'}\cup X_1^{''} = X_2^{'}\cup X_2^{''},$$

$$X_1^{'}\cap X_1^{''} = X_2^{'}\cap X_2^{''} = \varnothing$$

$$X_1^{'}\neq X_2^{'} \& X_1^{''}\neq X_2^{''},$$

графы, записанные в виде

$$(X_1^{'},\ X_1^{''};\ U)\ \textit{и}\ (X_2^{'},\ X_2^{''};\ U),$$

следует рассматривать лишь как изоморфные.

Матрица смежности графа Кёнига полностью определяется своей прямоугольной подматрицей, строки которой отвечают вершинам ***X'***, а столбцы – вершинам ***X''***, так, для графа на рисунке 8.1 это будет:

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

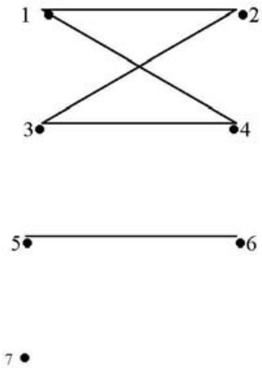


Рисунок 8.1

Задание графа Кёнига $L=\left(X', X''; U\right)$ равносильно заданию двух множеств X', X'' и отображения Γ, которое каждой вершине $x' \in X'$ относит подмножество (возможно пустое) $\Gamma x' \subseteq X''$ всех смежных с x' вершин $x'' \in X''$.

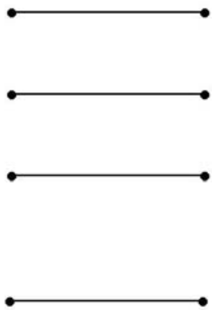


Рисунок 8.2

Граф Кёнига $L=\left(X', X''; U\right)$ называется *полным*, если каждая вершина множества X' смежна с каждой вершиной множества X'', иначе говоря, если $\forall x' \in X'\left(\Gamma x' = X''\right)$.

Граф Кёнига $K=(X,Y;U)$, в котором $|X|=|Y|=|U|=m\geq 1$ и никакие два ребра не смежны, называется *паросочетанием.*

Отображение Γ здесь фактически является взаимно однозначным соответствием между двумя множествами вершин X и Y.

Тема 9. КОМПОНЕНТЫ СВЯЗНОСТИ

Определение 1.

Граф $G=(X,U)$ связен, если любые его две вершины можно соединить простой цепью.

Определение 2.

Подграф G' графа G называется компонентой связности графа G, если: 1) G' связен, 2) G' обладает свойством максимальности, т.е. если G'' – некоторый другой связный подграф графа G и $G' \subset G''$, то графы G' и G'' совпадают.

Иными словами, *компонента связности* – это наибольший связный подграф данного графа.

На рисунке 9.1 показан граф $G=(X,U)$, содержащий две компоненты связности: G' с вершинами x_1, x_2, x_3 и G'' с вершинами x_4, x_5.

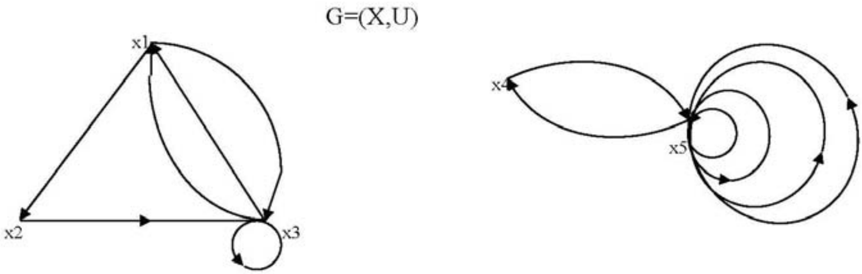


Рисунок 9.1

3. Компоненты связности. При помощи матриц смежности графов можно определить количество компонент связности. Для этого определим операцию элементарной склейки вершин на мультиграфе и выясним, как эта операция преобразует матрицу смежности.

Определение 3. Мультиграф $G' = (X', U')$ получен из мультиграфа $G= (X, U)$ при помощи операции элементарной склейки вершин x_i и x_j из X, если

1) $X' = (X \setminus (\{x_i\} \cup \{x_j\})) \cup \{z\}$, *где* $z \notin X$;

2) $(x_m, x_l, n) \in U'$ $(m, l \neq i, j)$ тогда и только тогда, когда $(x_m, x_l, n) \in U$, $(x_m, z, n) \in U'(m \neq i, j)$ тогда и только тогда, когда или $\boldsymbol{n = 2k}$ и $(x_m, x_i, k) \in U$, или $\boldsymbol{n = 2k} + 1$ и $(x_m, x_j, k) \in U$; $(z, z, n) \in U'$ *тогда и только тогда, когда или* $n = 4k$ и $(x_i, x_i, k) \in U$, *или* $n = 4k + 1$ *и*, $(x_j, x_j, k) \in U$, *или* $n=4k+2$ *и* $(x_i, x_j, k) \in U$, *или* $n=4k + 3$ и $(x_j, x_i, k) \in U$, $(z, x_m, n) \in U'$, *тогда и только тогда, когда* $n = 2k$ *и* $(x_i, x_m, k) \in U$ или $n = 2k+ 1$ и $(x_j, x_m, k) \in U$. Здесь n, k соответственно новый, старый номер ребра.

Иными словами, при склейке вершин x_i, и x_j склеиваются и концы дуг, совпадающие с x_i, и x_j, а сами дуги сохраняются.

Иллюстрацией может служить рисунок 9.2, где изображен граф G'' (рисунок 9.2, *б*), полученный элементарной склейкой вершин x_1, и x_2 графа G', изображенного на рисунке 9.2, *а*. Очевидно, склеивание двух вершин, лежащих в одной и той же компоненте связности, не изменяет количества компонент связности.

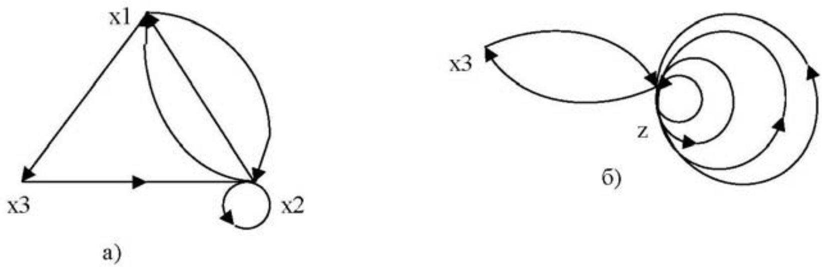


Рисунок 9.2

Обозначим $|X|$ через n. Перенумеруем множество вершин графа, полученного элементарной склейкой x_i и x_j следующим образом. Номера вершин, начиная с первого до i–1, сохраним. Номера вершин, начиная с i+1 до j–1, уменьшим на единицу, номера остальных вершин уменьшим на два, а вершине z присвоим номер $n-1$:

вершины	x_1, ...	x_{i-1},	x_{i+1},	x_{j-1},	x_{j+1},	...	x_n,	z
новые номера	1 ...	$i-1$	i ...	$j-2$	$j-1$	...	$n-2$	$n-1$

Обозначим через f(k) (k=1,2,..., n–2) старый номер вершины с новым номером k. Тогда матрица $\| a'_{ik} \|$ нового графа строится по матрице $\| a'_{ik} \|$ первоначального графа по формулам:

$$a'_{lk} = a_{f(l)f(k)} (l,k \le n-2),\quad a'_{n-1k} = a_{if(k)} + a_{jf(k)} (k \le n-2),$$

$$a'_{\ln-1} = a_{f(l)i} + a_{f(l)j} (l \le n-2),\quad a'_{n-1n-1} = a_{ii} + a_{ji} + a_{jj} + a_{ij}.$$

Например, матрицы смежности графов G и G' (см. рисунок 9.1, а, б) есть соответственно

$$A = \begin{Vmatrix} 0 & 2 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{Vmatrix} \quad \text{и} \quad A' = \begin{Vmatrix} 0 & 1 \\ 1 & 4 \end{Vmatrix}.$$

Заметим еще, что если в матрице смежности мультиграфа отличны от нуля лишь элементы, стоящие на главной диагонали, то число компонент связности равно числу вершин мультиграфа, т.е. размеру матрицы, так как такой мультиграф содержит только петли. На основании этого сформулируем алгоритм подсчета числа компонент связности по матрице мультиграфа.

Алгоритм.

ш.1. Найти ненулевой элемент матрицы смежности, не стоящий на главной диагонали. Если он существует, перейти ш.2, если нет, то перейти к ш.3.

ш.2. Произвести над матрицей операцию, отвечающую склейке вершин x_i и x_j, перейти к ш.1.

ш.3. Подсчитать количество p строк матрицы, содержащих ненулевые элементы на главной диагонали. Результат: количество компонент связности мультиграфа равно p.

Задание:

1. Определить число компонент связности для графа своего варианта с помощью алгоритма, рассмотренного в данной теме.
2. Исследовать алгоритм определения количества компонент связности для класса неориентированных графов.
3. Разработать алгоритм определения количества компонент связности для компьютерной обработки.

Тема 10. ОПТИМАЛЬНЫЕ ПОТОКИ В ТРАНСПОРТНЫХ / ИНФОРМАЦИОННЫХ СЕТЯХ

Теория транспортных и информационных сетей возникла при решении задач, связанных с организацией перевозки грузов и передачей информации. Тем не менее, понятие потока на транспортной сети и алгоритм нахождения потока наибольшей величины, критерий существования потока, насыщающего выходные дуги сети, оказались плодотворными для многих других прикладных и теоретических вопросов комбинаторного характера.

Введём основные понятия теории.

Транспортная/информационная сеть

Транспортная сеть T есть совокупность двух объектов:

1. Связного графа $G=(V,U)$, в котором отсутствуют петли и существует одна и только одна вершина $v_0 \in V$, такая, что множество $\Gamma^{-1}v_0 = \varnothing$, а также существует одна и только одна вершина $z \in V$, такая, что множество $\Gamma z = \varnothing$.

Вершина v_0 называется ***исток*** сети, вершина z называется – ***сток***.

2. Целочисленной неотрицательной функции c(u), заданной на множестве T дуг графа G.

Вершина v_0 называется входом сети T, вершина z – выходом. Значение функции c(u) на дуге $u \in U$ называется пропускной способностью дуги.

Поток на транспортной/информационной сети

Пусть U_x^- – множество дуг, заходящих в вершину $\boldsymbol{x}$, а U_x^+ – множество дуг, выходящих из вершины $\boldsymbol{x}$.

Целочисленная неотрицательная функция φ(u), заданная на множестве U дуг графа G, называется потоком, если она удовлетворяет следующим свойствам:

1. $\sum\limits_{u \in U_v^-} \varphi(u) = \sum\limits_{u \in U_v^+} \varphi(u) \;\; (v \neq v_0, \;\; v \neq z),$

2. $\varphi(u) \leq c(u)$.

Задача о максимальном потоке

Группу задач топологического анализа составляют задачи на распределение сетевых потоков. Это задача о максимальном потоке, в содержании которой оказываются взаимосвязанными топология сети, пропускные способности каналов связи или транспортных коммуникаций и распределение сетевых потоков. Подбирая определенным образом значение сетевых потоков и исходя из заданного распределения пропускных способностей каналов, можно получить единственно возможное максимальное для заданной топологии значение общего суммарного потока в сети. В этом случае, очевидно, ресурсы сетевой связи используются наиболее полно и эффективно.

Задача о максимальном потоке формулируется следующим образом: пусть задано исходное распределение потоков по дугам графа, отображающего топологическую структуру сети, а также пропускные способности дуг. Необходимо найти максимально возможное для данной сети значение суммарного потока между источниками и стоками, т.е. определить, как увеличить поток, если он не достиг этого значения. Применительно к рассматриваемой задаче используется одно из важных положений теории потоков, которое сформулировано и доказано в виде теоремы Фордом и Фалкерсоном. Согласно этой теореме максимально возможное значение суммарного потока на конечных дугах равно минимальной пропускной способности выбранного разреза. При этом под пропускной способностью разреза понимается сумма пропускных способностей дуг, образующих разрез.

В символической форме записи соотношение, отражающее содержание теоремы Форда–Фалкерсона, выглядит следующим образом:

$$\sum_{i\in B}\sum_{j\in \widetilde{B}}\varphi(v_i,v_j)=\sum_{i\in B}\sum_{j\in \widetilde{B}}c(v_i,v_j),$$

где $\varphi(v_i,v_j)$ – значение потока по дугам заданного графа; $c(v_i,v_j)$ – пропускная способность дуги; B – множество вершин подграфа, образующих разрез; $\widetilde{B}$ – дополнение множества B до множества ***V***. Множество В вершин подграфа, образующих разрез, всегда содержит вершину z (сток) и никогда не содержит вершину v_0 (исток). Разрез ***R(G)*** сети включает множество дуг $U' \subset U$ сети, исходящих из вершин множества $\widetilde{B}$ и входящих в вершины множества B.

Доказательство теоремы Форда–Фалкерсона строится методом от противного на следующих предположениях: граф имеет две характерные вершины – *исток* и *сток*, а разрез ***R(G)*** вершины графа делит на два взаимодополняющих множества B *и* $\widetilde{B}$.

Допустим, что на графе задан максимальный поток Ф, а сток z не отделен от множества вершин $\widetilde{B}$ разрезом, т.е. $z\in\widetilde{B}$. Тогда из определения разреза и при данном условии следует, что существует хотя бы один путь из истока V_0 в сток – вершину *z,* для которого должны выполняться условия: для прямых дуг пути (V_0 – Z) $\varphi(v_i, v_j) < c(v_i, v_j)$; для обратных $\varphi(v_i, v_j) > 0$.

Это свидетельствует о том, что поток на сети не является максимальным и его величину можно увеличить, насыщая отдельные дуги по путям, идущим от истока к стоку.

Иначе, если задан разрез, то он своими дугами однозначно определяет максимально возможный, проходящий через них поток. Очевидно, разрез (v_0–z) минимальной общей пропускной способностью дуг задает максимально возможный для данного графа поток x_0 к z.

Алгоритм Форда–Фалкерсона. Для транспортной/информационной сети важно найти такие минимальные разрезы и оценить соответствующие им значения максимального потока. Это можно осуществить с помощью алгоритма Форда–Фалкерсона.

Принцип, лежащий в основе алгоритма Форда–Фалкерсона, заключается в том, чтобы найти все возможные насыщенные пути (цепи), ведущие от v_0 к z. С этой целью последовательно, начиная с вершины $x_{0,}$ просматриваются сначала все смежные ей $\{v_i\}$ вершины. Из множества дуг $\{(v_0, v_{i)}\}$, соединяющих v_0 с $\{ v_i \}$, выбирают одну, у которой значение потока ближе всех подходит к значению насыщения. Помечают вершину v_i знаком, показывающим, что она была просмотрена, и приписывают ей величину δ, на которую можно увеличить поток по дуге, ведущей в эту вершину. Затем просматривают все последующие смежные с v_i вершины $\{ v_j \}$ и останавливаются на той, в которую ведёт дуга с потоком, ближайшим к значению насыщения. По этой дуге переходят в соответствующую вершину v_j. Делают пометки вершин и идут далее в направлении вершины z. Если путь, на котором будут отмечены все пройденные вершины, приведёт в вершину z, то это говорит о том, что найден один из путей, наиболее близкий к насыщению. Нужно довести поток по нему до насыщения, увеличивая тем самым поток на графе. Значение, на которое можно увеличить поток, находится как минимальное δ из множества отмеченных значений в пройденных по данному пути вершинах. Для выяснения вопроса, является ли полученный таким образом поток максимальным или нет, необходимо просмотреть все другие возможные пути, ведущие от x_0 к z. Для этого необходимо возвратиться в x_0 и повторить описанные выше действия, но идти следует по еще не помеченным вершинам. В результате выполнения этих действий можно столкнуться с двумя вариантами: 1) пройденный путь снова приводит от x_0 к z, т.е. удается найти ненасыщенные пути и, значит, можно увеличить потоки на их дугах; 2) придя в некоторую вершину, обнаружить, что все смежные с ней вершины помечены и, следовательно, больше путей, ведущих к z, нет. Это указывает на то, что на сети получен максимальный поток.

Согласно теореме Форда и Фалкерсона значение полученного максимального потока можно вычислить, выделив дуги разреза ***R(G)*** с минимальной пропускной способностью и просуммировав их пропускные способности.

Пример решения задачи о максимальном потоке даётся в программе ford2, записанной на диске для специальностей 220300 / 230104 (САПР) и 210100/220201 (управление и информатика в технических системах).