

## ОПРЕДЕЛЕНИЕ ЧИСЛОВЫХ ХАРАКТЕРИСТИК ГРАФОВ

### Цель

Знакомство с понятием графа, видами графов, способами их представления, основными числовыми характеристиками, алгоритмами анализа и синтеза.

Получение навыков представления графов для вычисления их числовых характеристик и вычисление некоторых из них.

### Задание

1. Выбрать вариант задания из [Приложения 1](#).

2. Составить программу(ы) на Python, выполняющую задание.

Требования к оформлению программы:

- Текстовые пояснения действия каждой подпрограммы, функции, процедуры, оператора.

3. Составить отчет по проделанной работе.

Требования к содержанию:

- Титульный лист,
- Цель работы,
- Задание,
- Математическая постановка задачи,
- Описание алгоритма решения,
- Блок-схемы алгоритмов программы, оформленные по ЕСПД,
- Листинг написанных программ,
- Описание факторного эксперимента для тестирования программы,
- Таблицы тестирования,
- Заключение.

Требования к оформлению отчёта:

- Все страницы, кроме первой (титульный лист), должны быть пронумерованы в верхнем правом углу.
- В верхнем колонтитуле каждой страницы, кроме первой, разместить с выравниванием влево фамилию и инициалы студента и надпись «Лабораторная работа № 5».
- Присвоить файлу имя «Лабораторная работа № 5».

Порядок сдачи отчёта:

- Сохранить файл отчёта в соответствующей папке Google Диска.
- Оповестить преподавателя через почту о созданном отчете и прикрепить отчет.
- Получить замечания по документу, в соответствии с замечаниями внести изменения в документ. Оповестить преподавателя ответом на его письмо.

## Введение

Понятие и теория графов начались с поиска методов решений головоломок и игр (таких как, задача о кенигсбергских мостах и игра Гамильтона) и развились до отдельной математической дисциплины (одной из ветвей дискретной математики).

Параллельное развитие графовых представлений (моделей) в рамках решения теоретических и прикладных задач привело к множественности терминов, обозначающих графы:

- «сети» – в электронике,
- «молекулярные структуры» – в химии,
- «структуры» – в гражданском строительстве,
- «социограммы» – в социологии и экономике,
- «дорожные карты» – в управлении (менеджменте),
- и т. д.

В настоящее время графы применяются для моделирования (структуры, поведения) различных объектов, представления физических, информационных и управляющих процессов, решения алгебраических, логических и комбинаторных задач, возникающих во многих сферах деятельности:

- логистический анализ,
- менеджмент проектов,
- проектирование и оптимизация организационных структур,
- разводка печатных плат,
- расчёты параметров и режима электрических цепей,
- расчёты картины электромагнитного поля,
- оценка надёжности электроснабжения (например, по критерию  $N - 1$ ),
- оценка эффективности использования ресурсов вычислительных систем,
- организация больших массивов информации,
- построение трансляторов и компиляторов языков программирования,
- оценка быстродействия систем передачи и обработки данных,
- и т. д.

Считается, что столь разнообразное применение графов основано на их естественном (сущностном) описании (представлении, объяснении) реальности.

## Понятие и определения графа

Граф понимается как множество точек (вершин, узлов), которые соединяются множеством линий (рёбер, дуг, отрезков, связей).

Можно дать несколько определений графа, например:

1. Графом  $G$  называется пара множеств  $V$  и  $E$ , где  $V(G)$  – непустое множество объектов некоторой природы, называемых вершинами графа  $G$ , а  $E(G)$  – подмножество двухэлементных подмножеств множества  $V(G)$ , называемых рёбрами графа  $G$ .

2. Графом  $G$  называется пара  $V$  и  $E$ , где  $V(G)$  – множество вершин, а  $E$  – множество рёбер – есть подмножество множества  $V^2 / \sim$  классов эквивалентности, на которые множество

$$V^2 = \{(v, w), v \neq w\}$$

разбивается отношением эквивалентности

$$(v_1, w_1) \sim (v_2, w_2) \Leftrightarrow (v_1, w_1) = (v_2, w_2) \text{ или } (v_1, w_1) = (w_2, v_2).$$

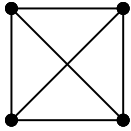
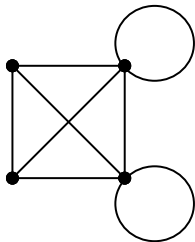
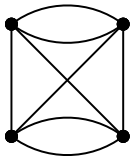
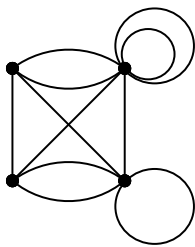
3. Графом  $G$  называется тройка множеств  $(V, E, P)$ , где  $V$  – множество вершин,  $E$  – множество объектов некоторой природы, отличной от природы вершин, называемых рёбрами,  $P$  – инцидентор, сопоставляющий каждому ребру  $e \in E$  пару граничных вершин  $v$  и  $w$  из  $V$ .

Обычно граф изображают диаграммой: вершины – точками, рёбра – линиями.

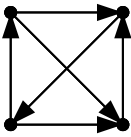
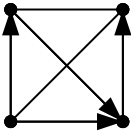
## Типы графов

Основные типы графов представлены в таблице 1.

Таблица 1 – Основные типы графов

Наименование	Изображение (пример)	Определение	Математическое выражение	Примечания
1	2	3	4	5
Простой		Совокупность двух множеств – непустого множества вершин $V$ и множества рёбер $E$ , состоящего из неупорядоченных пар различных элементов множества $V$ .	$G(V, E) = \langle V, E \rangle$ $V \neq \emptyset$ $E \subseteq V \times V$ $\{v, v\} \notin E$ $v \in V$	В графе запрещены петли – рёбра, у которых концевые вершины совпадают ( $e = \{v, v\}$ ).
Псевдограф		Совокупность двух множеств – непустого множества вершин $V$ и множества рёбер $E$ , состоящего из неупорядоченных пар элементов множества $V$ .	$G(V, E) = \langle V, E \rangle$ $V \neq \emptyset$ $E \subseteq V \times V$ $\{v, v\} \in E$ $v \in V$	В графе разрешены петли.
Мультиграф		Совокупность двух множеств – непустого множества вершин $V$ и мультимножества рёбер $E$ , состоящего из неупорядоченных пар различных элементов множества $V$ .	$G(V, E) = \langle V, E \rangle$ $V \neq \emptyset$ $E \subseteq V \times V$ $\{v, v\} \notin E$ $v \in V$	Множество $E$ является семейством – содержит одинаковые элементы (кратные рёбра).  В графе запрещены петли.
Псевдомультиграф		Совокупность двух множеств – непустого множества вершин $V$ и мультимножества рёбер $E$ , состоящего из неупорядоченных пар элементов множества $V$ .	$G(V, E) = \langle V, E \rangle$ $V \neq \emptyset$ $E \subseteq V \times V$ $\{v, v\} \notin E$ $v \in V$	Множество $E$ является семейством – содержит одинаковые элементы (кратные рёбра).  В графе разрешены петли.

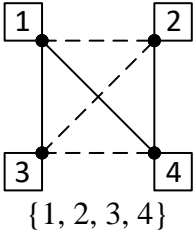
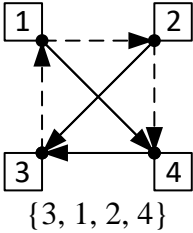
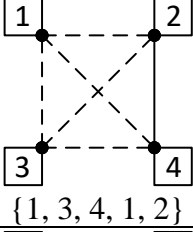
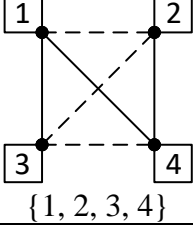
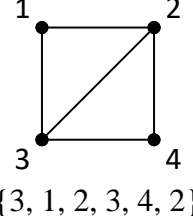
Продолжение таблицы 1

1	2	3	4	5
Ориентированный (орграф)		Совокупность двух множеств – непустого множества вершин $V$ и множества дуг $E$ , состоящего из упорядоченных пар различных элементов множества $V$ .	$G(V, E) = \langle V, E \rangle$ $V \neq \emptyset$ $E \subseteq V \times V$ $\langle \{v_1, v_2\}, < \rangle \in E$ $v \in V$	Вершина $v_1$ дуги $\{v_1, v_2\}$ является началом, может обозначаться как $init(e)$ .  Вершина $v_2$ дуги $\{v_1, v_2\}$ является концом, может обозначаться как $ter(e)$ .
Смешанный		Совокупность трёх множеств – непустого множества вершин $V$ и множества дуг $E$ или упорядоченных пар различных элементов множества $V$ и множества рёбер $U$ неупорядоченных пар различных элементов множества $V$ .	$G(V, E, U) = \langle V, E, U \rangle$ $V \neq \emptyset$ $\langle \{v_1, v_2\}, < \rangle \in E$ $\{v_3, v_4\} \in U$ $v \in V$	Граф содержит как неориентированные рёбра, так и ориентированные дуги.

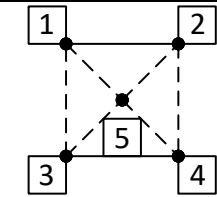
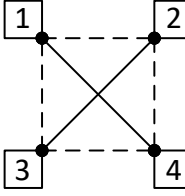
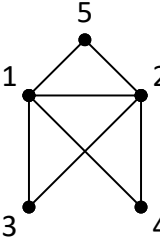
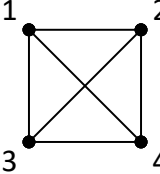
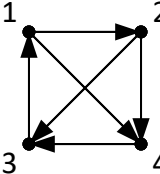
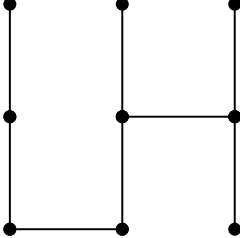
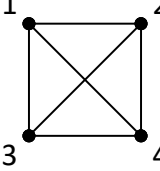
## Дополнительные понятия и определения

Некоторые дополнительные понятия и определения теории графов представлены в таблице 2.

Таблица 2 – Дополнительные понятия и определения теории графов

Понятие (термин)	Определение	Изображение (пример)
1	2	3
Маршрут (путь)	Конечная последовательность вершин, в которой каждая вершина, кроме последней, соединена ребром со следующей в последовательности вершиной.	 $\{1, 2, 3, 4\}$
Оrientированный маршрут (путь)	Конечная последовательность вершин, в которой каждая вершина, кроме последней, соединена дугой со следующей в последовательности вершиной.	 $\{3, 1, 2, 4\}$
Цепь	Маршрут без повторяющихся рёбер.	 $\{1, 3, 4, 1, 2\}$
Простая цепь	Цепь без повторяющихся вершин	 $\{1, 2, 3, 4\}$
Эйлерова цепь	Путь, проходящий по всем рёбрам графа и притом только по одному разу	 $\{3, 1, 2, 3, 4, 2\}$

Продолжение таблицы 2

1	2	3
Цикл	Цепь, в которой первая и последняя вершины совпадают.	 $\{1, 3, 5, 4, 2, 5, 1\}$
Простой цикл	Цикл без повторяющихся вершин (за исключением начальной и конечной).	 $\{1, 2, 4, 3, 1\}$
Эйлеров цикл	Цикл, проходящий через каждое ребро графа ровно по одному разу	 $\{3, 2, 1, 5, 2, 4, 1, 3\}$
Связный граф	Граф, в котором для любых вершин $v$ и $u$ есть путь из $v$ в $u$ .	
Сильносвязный граф	Орграф, в котором для любых вершин $v$ и $u$ есть ориентированный путь из $v$ в $u$ .	
Дерево	Связный граф, не содержащий циклов.	
Полный граф	Граф, в котором любые две различные вершины соединены ребром.	

Продолжение таблицы 2

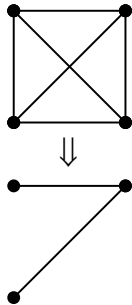
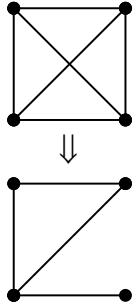
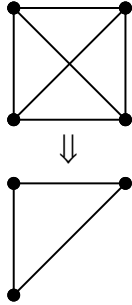
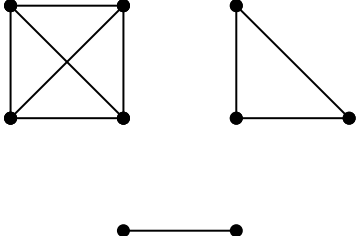
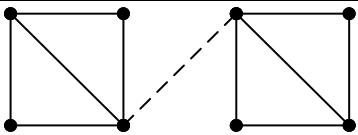
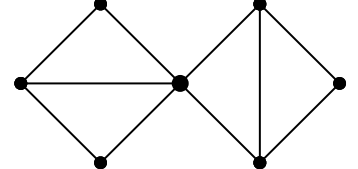
1	2	3
Двудольный граф	Граф, множество вершин которого можно разбить на два не пересекающихся подмножества $V_1$ и $V_2$ так, что любое ребро соединяет какую-либо вершину из $V_1$ с вершиной из $V_2$ .	
Полный двудольный граф	Двудольный граф, в котором каждая вершина одного подмножества соединена ребром с каждой вершиной другого подмножества.	
k-дольный граф	Граф, в котором множество вершин можно разбить на k непересекающихся подмножеств $V_1, V_2, \dots, V_k$ , так, что множество рёбер, соединяющих вершины одного и того же подмножества, будет пустым.	
Планарный граф	Граф, который можно изобразить диаграммой на плоскости без пересечения рёбер.	

Продолжение таблицы 2

1	2	3
Взвешенный граф	Граф, в котором каждому ребру поставлено в соответствие некоторое число (вес).	
Изоморфный граф	<p>Два неориентированных невзвешенных графа <math>G = \langle V_G, E_G \rangle</math> и <math>H = \langle V_H, E_H \rangle</math> являются изоморфными друг другу <math>G \simeq H</math>, если существует биекция между множествами вершин графов <math>f: V_G \rightarrow V_H</math> такая, что любые две вершины <math>u</math> и <math>v</math> графа <math>G</math> смежны тогда и только тогда, когда вершины <math>f(u)</math> и <math>f(v)</math> смежны в графе <math>H</math>.</p> <p>Для определения изоморфизма ориентированных и/или взвешенных графов на биекция должна содержать дополнительные ограничения на сохранение ориентации дуг и значений весов.</p>	



Продолжение таблицы 2

1	2	3
Подграф	Граф, состоящий из подмножества вершин исходного графа и подмножества рёбер исходного графа, соединяющих эти вершины.	
Остовный подграф	Подграф, имеющий такое же количество вершин, как и основной граф, но множество рёбер подграфа является подмножеством множества рёбер исходного графа.	
Порожденный подграф	Подграф, состоящий из подмножества вершин исходного графа и всех рёбер исходного графа, соединяющих эти вершины.	
Компонента связности (связная компонента, компонента)	Максимальный связный подграф графа	 <p>(три компоненты связности)</p>
Мост	Ребро, удаление которого из графа увеличивает число компонент	
Шарнир	Узел, удаление которого из графа увеличивает число компонент	

### Способы представления графов

Распространены четыре способа представления графов:

1. Матрица смежности,
2. Матрица инцидентности,
3. Список смежности,
4. Список инцидентности.

Рассмотрим эти способы на примере неориентированного связного графа, изображённого на рисунке 1.

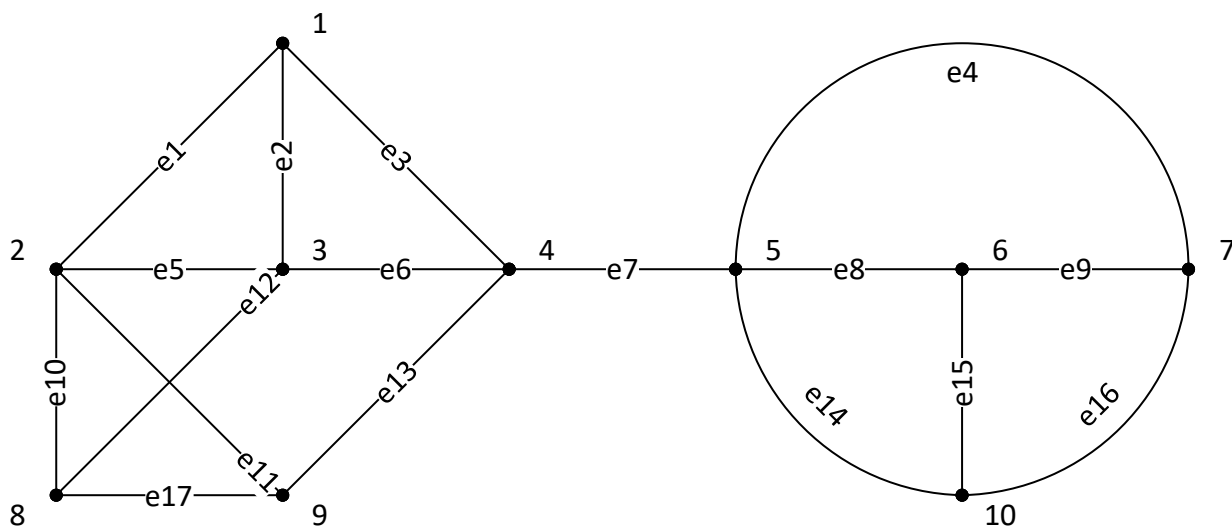


Рисунок 1 – Неориентированный связный граф

1. Матрица смежности графа  $G$  с конечным числом вершин  $n$  (пронумерованных числами от 1 до  $n$ ) – это квадратная целочисленная матрица  $A$  размера  $n \times n$ , в которой значение элемента  $a_{i,j}$  равно числу рёбер из  $i$ -ой вершины графа в  $j$ -ю вершину.

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

**Примечание:** Матрица смежности пустого графа (не содержащего ни одного ребра) состоит из одних нулей.

**Примечание:** Матрица смежности неориентированного графа симметрична относительно главной диагонали ( $\forall i,j: a_{i,j} = a_{j,i}$ ).

**Примечание:** Матрица смежности простого графа (не содержащего петель и кратных рёбер) является бинарной матрицей (её элементы равны или 0, или 1) и содержит нули на главной диагонали.

**Примечание:** В случае неориентированного графа с петлями, петля (ребро из  $i$ -ой вершины в саму себя) считается за два ребра, то есть значение диагонального элемента  $a_{i,i}$  равно удвоенному числу петель вокруг  $i$ -ой вершины.

**Примечание:** Использование матрицы смежности рекомендуется только для неразрезанных графов (с числом рёбер сопоставимым по порядку или превышающим число вершин). Если граф разрежен, то большая часть памяти, занимаемая матрицей смежности, будет тратиться на хранение нулей.

2. Матрица инцидентности графа  $G$  с конечным числом вершин  $n$  (пронумерованных числами от 1 до  $n$ ) и рёбер  $m$  (пронумерованных числами от 1 до  $m$ ) – это целочисленная матрица  $I$  размера  $n \times m$ , в которой столбцы соответствуют рёбрам, строки – вершинам. Ненулевое значение в ячейке матрицы указывает на связь между вершиной и ребром (инцидентность).

$$I(G) = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{vmatrix}$$

**Примечание:** В каждом столбце неориентированного связного графа, не содержащего петель, стоят только две единицы.

В каждом столбце неориентированного графа, не содержащего петель, стоят или две единицы, или ни одной.

3. Список смежности – способ представления графа  $G$  в виде коллекции (массива, вектора) списков вершин. Каждой вершине графа соответствует список смежных с ней вершин.

$$A(G) = \begin{vmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \\ a_{10} \end{vmatrix}, \text{ где } \begin{matrix} a_1 = \{2, 3, 4\} \\ a_2 = \{1, 3, 8, 9\} \\ a_3 = \{1, 2, 4, 8\} \\ a_4 = \{1, 3, 5, 9\} \\ a_5 = \{4, 6, 7, 10\} \\ a_6 = \{5, 7, 10\} \\ a_7 = \{5, 6, 10\} \\ a_8 = \{2, 3, 9\} \\ a_9 = \{5, 4, 8\} \\ a_{10} = \{5, 6, 7\} \end{matrix}$$

**Примечание:** Компактный способ представления разреженных графов. Размер занимаемой памяти равен  $O(|V| + |E|)$ .

**Примечание:** Простая реализация базовых алгоритмов обходов графа в ширину и глубину (прямой доступ к вершинам, смежным текущей).

4. Список инцидентности – способ представления графа  $G$  в виде коллекции (массива, вектора, списка) рёбер:

$$I(G) = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} \\ \hline \end{array}$$

Каждому ребру  $a_i$  графа соответствует строка, содержащая две вершины (начало и конец):

$$\begin{aligned} a_1 &= \{1, 2\} \\ a_2 &= \{1, 3\} \\ a_3 &= \{1, 4\} \\ a_4 &= \{5, 7\} \\ a_5 &= \{2, 3\} \\ a_6 &= \{3, 4\} \\ a_7 &= \{4, 5\} \\ a_8 &= \{5, 6\} \\ a_9 &= \{6, 7\} \\ a_{10} &= \{2, 8\} \\ a_{11} &= \{2, 9\} \\ a_{12} &= \{3, 8\} \\ a_{13} &= \{4, 9\} \\ a_{14} &= \{5, 10\} \\ a_{15} &= \{6, 10\} \\ a_{16} &= \{7, 10\} \\ a_{17} &= \{8, 9\} \end{aligned}$$

**Примечание:** Наиболее компактный способ представления графов.  
Размер занимаемой памяти равен  $O(|E|)$ .

## Основные числовые характеристики графов

К основным числовым характеристикам графов можно отнести:

- число вершин,
- число рёбер,
- степени вершин,
- число мостов,
- число шарниров,
- число петель,
- число кратности рёбер,
- число простых контуров,
- число компонент связности,
- число независимых (состоящих из различных вершин и рёбер) маршрутов из  $v_i$  в  $v_j$ .
- цикломатическое число (наименьшее число рёбер, удаление которых приводит к графу без циклов),
- хроматическое число (наименьшее количество цветов, которыми можно раскрасить вершины (ребра) граф так, чтобы любые смежные вершины (ребра) были окрашены разными цветами),
- и другие.

## Основные свойства графов

К основным свойствам графов принято относить:




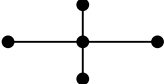
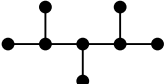

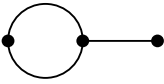

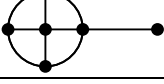
- связность (рёберная, вершинная, 1-связность,  $k$ -связность и т.д.) – граф называется  $k$ -вершинно связным, если в нём найдётся множество из  $k$  вершин, удаление которых делает граф несвязным, и никакое множество из меньшего количества вершин не обладает этим свойством,
- планарность (в общем случае) – укладка графа на заданной поверхности без пересечений рёбер,
- (полу)эйлеровость – существование в графе эйлера (пути) цикла,
- гамильтоновость – существование в графе гамильтонова цикла (простого цикла содержащего все вершины этого графа),
- раскрашиваемость – возможность разметки вершин или рёбер графа в соответствии с установленным правилом,
- и другие.

## Распознавание графов

Под распознаванием графов понимается идентификация класса графа (отнесение конкретного графа к одному из определённых классов). Дополнительными задачами распознавания принято считать, например, оценку возможности перевода данного графа  $G$  (принадлежащего классу  $X$ ) в класс  $Y$  добавлением (удалением)  $k$  вершин (рёбер).

В теории графов рассматриваются различные классификации графов. Один из примеров классификации зависимости от значений числовых характеристик и соотношений между ними дан в таблице 3.

Таблица 3 – Пример классификация связных ненаправленных мультиграфов

Тип	Вид	Изображение	Соотношение значений числовых характеристик
1	2	3	4
Узел	Вырожденный граф		$ G  = \ G\  + 1$ $\ G\  = 0$
Цепи	Односвязная		$ G  = \ G\  + 1$ $\ G\  = 1$
	Многосвязная		$ G  = \ G\  + 1$ $\ G\  > 1$ $\forall v, 1 \leq \deg(v) \leq 2$
Деревья	Радиальные		$ G  = \ G\  + 1$ $\ G\  > 2$ $\exists! v, \deg(v) > 2$
	Древовидные		$ G  = \ G\  + 1$ $\ G\  > 4$ $\exists v \text{ и } \neg \exists! v, \deg(v) > 2$
Контурные	Одноконтурные		$ G  = \ G\ $ $\ G\  > 1$ $\forall v, \deg(v) = 2$
	Содержащие один контур		$ G  = \ G\ $ $\ G\  > 2$
	Многоконтурные		$ G  < \ G\ $ $\ G\  > 2$ $\neg \exists v, \deg(v) = 1$
	Содержащие множество контуров		$ G  < \ G\ $ $\ G\  > 2$

**Примечание:**  $|G|$  – число вершин графа  $G$ ,  
 $\|G\|$  – число рёбер графа  $G$ ,  
 $\deg(v)$  – степень вершины  $v$ .

## Операции над графами

Для выявления свойств графа может требоваться его преобразование к иному виду.

Основные одноместные операции преобразования графа:

- удаление (добавление) ребра  $E - e$  или вершины  $V - v$ ,
- стягивание ребра (отождествление пары смежных вершин).

Пусть  $e = xy$  есть ребро в графе  $G = (V, E)$ .  $G/e$  - граф, полученный из  $V - v$  стягиванием ребра  $e$  в новую вершину  $v_e$ , которая становится смежной со всеми соседями вершин  $x$  и  $y$ .

Формально  $G/e = (V', E')$  со множеством вершин  $V' = (V \setminus \{x, y\}) \cup \{v_e\}$  и множеством рёбер  $\{vw \in E \mid \{v, w\} \cap \{x, y\} = \emptyset\} \cup \{v_e w \mid xw \in E \setminus \{e\} \text{ или } yw \in E \setminus \{e\}\}$ ;

- подразбиение ребра (на пару).

Основные двуместные операции преобразования графов:

- соединение,
- сложение,
- умножение
- и другие.



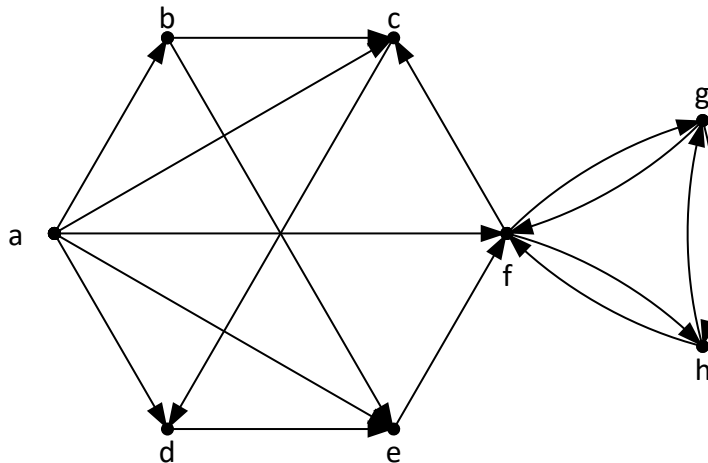
## Примеры некоторых основных алгоритмов анализа графов

**Примечание:** Кроме задач анализа графов существуют задачи их синтеза – построение графов с заданными числовыми характеристиками и структурными свойствами.

Далее представлено несколько основных алгоритмов анализа графов.

### Пример 1. Обход графа в ширину

Дано: Связный мультиорграф G.



Выяснить: Инцидентность вершины a вершине h (существование ориентированного маршрута из a в h) и из h в a.

Решение: Существование ориентированного маршрута определим, используя алгоритм обхода графа в ширину.

Описание алгоритма:

1. Поместить вершину, с которой начинается поиск, в изначально пустую очередь (посещённых вершин).
2. Поместить вершину, с которой начинается поиск, в множество достигнутых (посещённых) вершин.
3. Извлечь из начала очереди вершину u и пометить её как развёрнутую.
4. Если вершина u является конечной вершиной, то завершить поиск с результатом «успех».
5. Иначе, добавить в конец очереди все вершины смежные вершине u, которые ещё не развёрнуты и не находятся в очереди.
6. Если очередь пуста, то все вершины связного графа были просмотрены, следовательно, конечная вершина недостижима из начальной; завершить поиск с результатом «неудача».
7. Вернуться к п. 3.

**Примечание:** Алгоритм обхода графа в ширину можно применять не только для поиска маршрута из одной вершины в другую, но и для проверки графа на связность, или для вычисления числа компонентов связности графа.

## Реализация на Python:

```
#импорт класса очереди
from queue import Queue

#инициализация исходных данных

#граф G
#список вершин графа G
a, b, c, d, e, f, g, h = range(8)
#список смежности вершин графа G
N = [
    {b, c, d, e, f}, # a
    {c, e}, # b
    {d}, # c
    {e}, # d
    {f}, # e
    {c, g, h}, # f
    {f, h}, # g
    {f, g} # h
]

#начальная вершина
beg_node=a

#конечная (целевая) вершина
target_node=h

#инициализация начальных значений используемых переменных

#перечень вершин в виде строки
letters="abcdefgh"

#достигнутые (посещённые) вершины
visited_nodes={}

#очередь развёртываемых вершин
queue=Queue()

#поиск маршрута из начальной вершины в конечную (целевую)

#шаг 1. помещение начальной вершины в (пустую) очередь
queue.put(beg_node)

#шаг 2. помещение начальной вершины в множество, достигнутых
(посещённых) вершин
visited_nodes={beg_node}
```

```

#установка значения флага успешности поиска в ЛОЖЬ
find=False

#цикл поиска маршрута из начальной вершины в конечную
while (not find) and (not queue.empty()):

    #шаг 3. извлечение (возврат значения и удаление из очереди) вершины
    из начала очереди queue и присвоение её значения переменной <u>
    u=queue.get()

    #вывод на дисплей информации о разворачиваемой вершине
    print("Обход вершины",letters[u])

    #шаг 4. если вершина <u> является конечной, то завершить поиск с
    результатом «успех»
    if u==target_node:
        #установка значения флага успешности поиска в ИСТИНА
        find=True
        #вывод на дисплей информации об успешном завершении поиска
        print("Поиск успешно завершён")

    #шаг 5. иначе, в конец очереди добавляются все преемники вершины
    <u>, которые ещё не развёрнуты и не находятся в очереди
    else:
        #цикл записи всех преемников вершины <u> в очередь и в множество
        достигнутых (просмотренных вершин)
        for node in N[u]:
            print(" -- Просмотр дуги", letters[node])
            #если преемник вершины <u> не включён в множество достигнутых
            (посещенных) узлов
            if not(node in visited_nodes):
                #включение преемника вершины <u> в множество достигнутых
                (посещенных) узлов
                visited_nodes.add(node)
                #помещение преемника вершины <u> в очередь
                queue.put(node)
                #вывод на дисплей информации о помещении преемника вершины
                <u> в очередь
                print("    ---- Узел", letters[node],"добавили в очередь")

    #шаг 6. если очередь пустая, то все вершины графа просмотрены, а
    конечная (целевая) вершина не достигнута из начальной; завершить поиск
    с результатом «неудача»
    #если очередь пустая
    if queue.empty():
        #вывод на дисплей информации о том, что вершина не найдена
        print("Вершина ",letters[u]," не найдена")

#шаг 7. Вернуться к п. 3.

```

Блок-схема алгоритма по ЕСПД: представлена на рисунках 2-4.



Рисунок 2 – Блок-схема алгоритма по ЕСПД

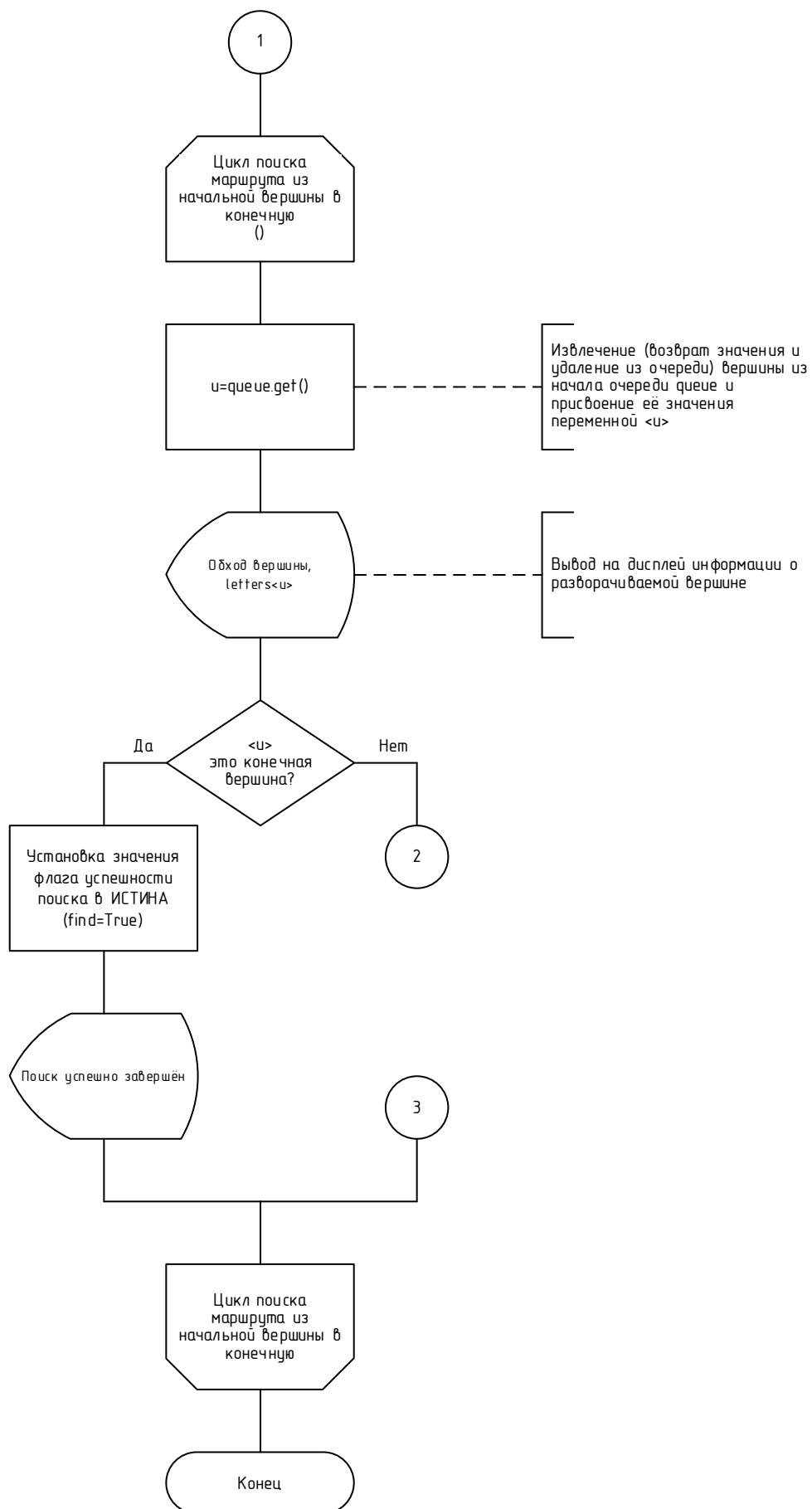


Рисунок 3 – Блок-схема алгоритма по ЕСПД (продолжение)

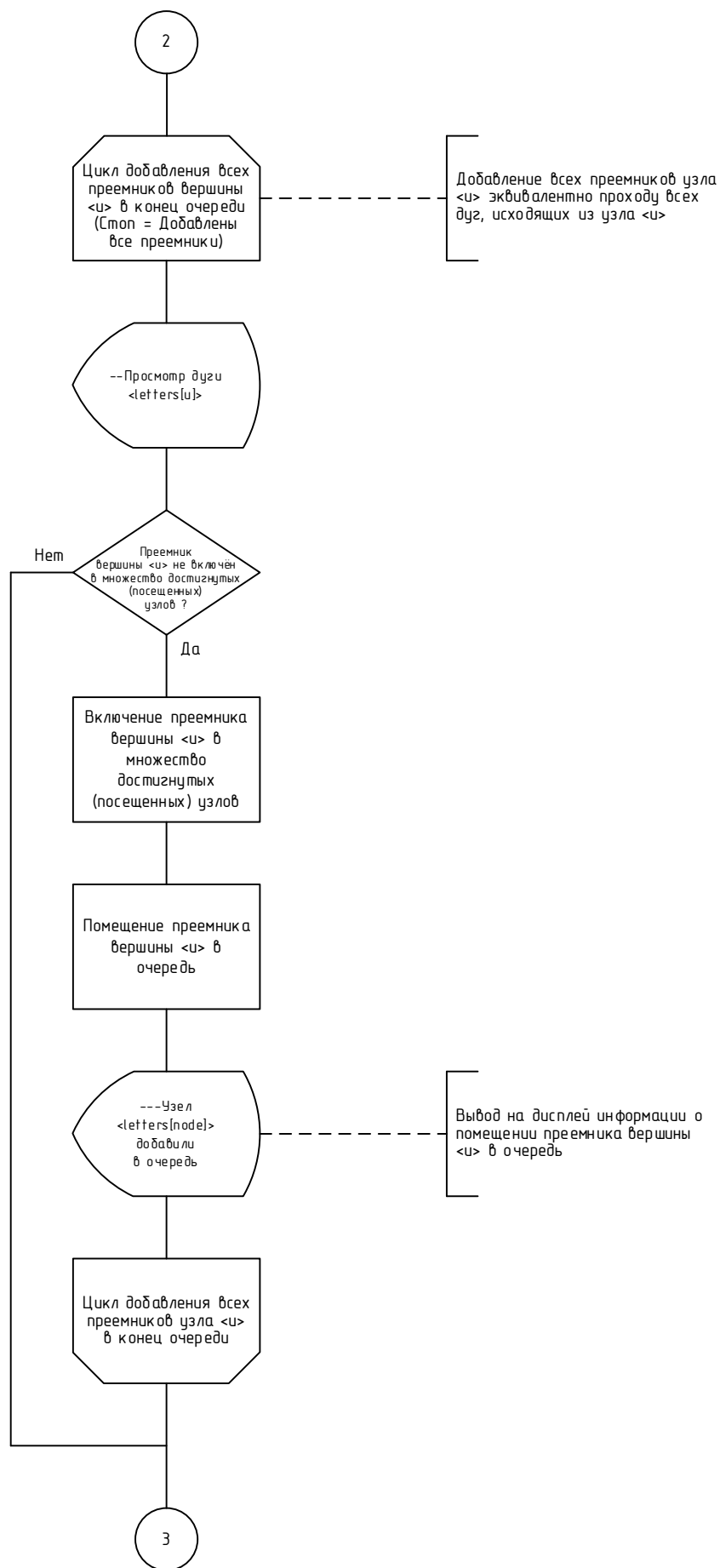


Рисунок 4 – Блок-схема алгоритма по ЕСПД (окончание)

Результат действия программы при поиске пути из а в h:

```
Обход вершины а
-- Просмотр дуги b
---- Вершина b добавлена в очередь
-- Просмотр дуги c
---- Вершина c добавлена в очередь
-- Просмотр дуги d
---- Вершина d добавлена в очередь
-- Просмотр дуги e
---- Вершина e добавлена в очередь
-- Просмотр дуги f
---- Вершина f добавлена в очередь
Обход вершины b
-- Просмотр дуги c
-- Просмотр дуги e
Обход вершины c
-- Просмотр дуги d
Обход вершины d
-- Просмотр дуги e
Обход вершины e
-- Просмотр дуги f
Обход вершины f
-- Просмотр дуги c
-- Просмотр дуги g
---- Вершина g добавлена в очередь
-- Просмотр дуги h
---- Вершина h добавлена в очередь
Обход вершины g
-- Просмотр р дуги f
-- Просмотр дуги h
Обход вершины h
Поиск успешно завершён
```

Результат действия программы при поиске пути из h в a:

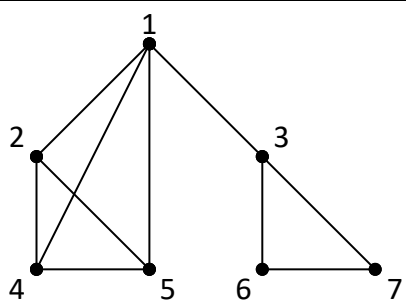
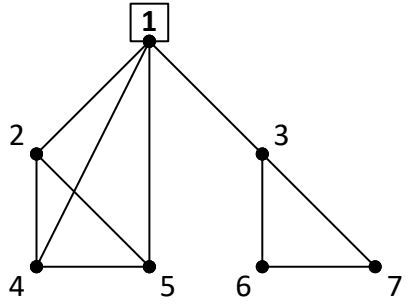
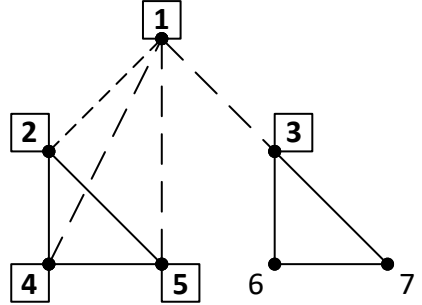
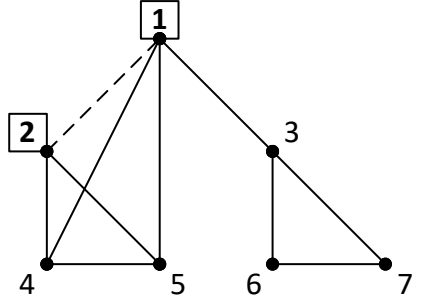
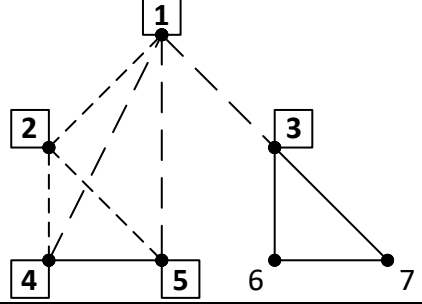
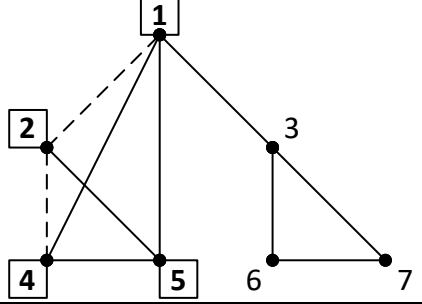
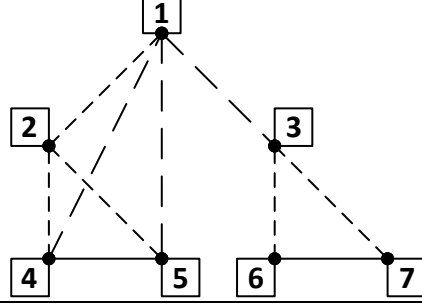
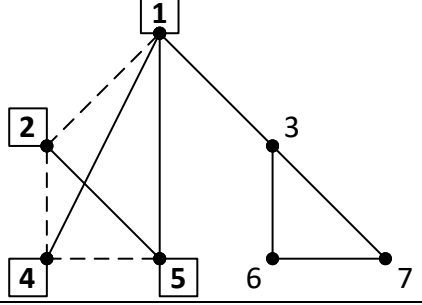
```
Обход вершины h
-- Просмотр дуги f
---- Узел f добавили в очередь
-- Просмотр дуги g
---- Узел g добавили в очередь
Обход вершины f
-- Просмотр дуги c
---- Узел c добавили в очередь
-- Просмотр дуги g
-- Просмотр дуги h
Обход вершины g
-- Просмотр дуги f
-- Просмотр дуги h
Обход вершины c
-- Просмотр дуги d
---- Узел d добавили в очередь
Обход вершины d
-- Просмотр дуги e
---- Узел e добавили в очередь
Обход вершины e
-- Просмотр дуги f
Вершина e не найдена
```



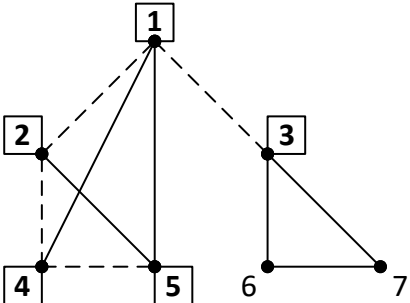
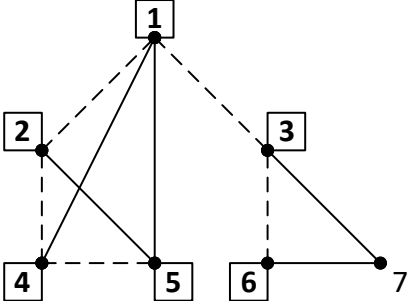
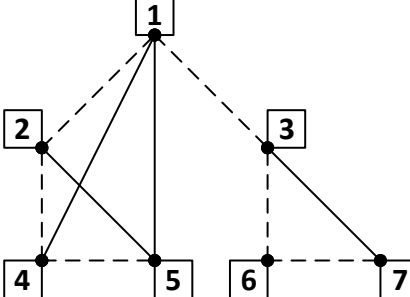
## Пример 2. Обход графа в глубину

В данном примере ограничимся сопоставлением результатов шагов обхода графа (см. таблицу 4) в ширину и глубину при проверке его связности.

Таблица 4 – Сопоставление результатов шагов поиска в ширину и глубину

Номер шага	Поиск в ширину	Поиск в глубину
1	2	3
0		
1		
2		
3		
4		

Продолжение таблицы 4

1	2	3
5	-	
6	-	
7	-	

**Примечание:** Просмотренные вершины помечаются рамкой, рёбра - пунктиром.

**Примечание:** В качестве примера реализации на Python алгоритма поиска в глубину можно использовать адаптацию кода из примера 1 с использованием класса стека LifoQueue, содержащегося модуле queue.

### Пример 3. Поиск эйлерова цикла на графе. Алгоритм Флёри

Дано: Эйлеров граф  $G$ .

Найти: Эйлеров цикл.

Решение: Поиск эйлерова цикла в эйлеровом графе  $G$ , понимается как нумерация рёбер графа числами  $1, 2, \dots, m$  (где  $m = |E|$ ), указывающими на то, каким по счёту каждое ребро входит в эйлеров цикл. Нумерация рёбер (обход графа) выполняется в два основных этапа:

#### 1. Начало

- выберем произвольную вершину  $u$ ;
- присвоим произвольному ребру  $(u, v)$ , не являющемуся мостом, номер 1;
- вычеркнем (отметим, как включённое в эйлеров цикл) ребро  $(u, v)$ ;
- перейдём в вершину  $(v)$ .

#### 2. Поиск

- пусть  $w$  – вершина, в которую был выполнен переход в результате выполнения предыдущего шага, и  $k$  – номер, присвоенный некоторому ребру на этом шаге.
- выберем другое ребро, инцидентное вершине  $w$ , причём мост выберем только в том случае, когда нет других возможностей,
- присвоим выбранному ребру номер  $k + 1$ ,
- вычеркнем это ребро.

Решение достигается (условие завершения), когда все рёбра графа пронумерованы (помечены, «вычеркнуты»).

**Примечание:** На каждом шаге алгоритма Флёри необходимо проверять является ли выбранное ребро мостом или нет, то есть не приводит ли удаление выбранного ребра к разбиению графа на две связные компоненты (не считая изолированных вершин).

**Примечание:** Алгоритм Флёри может быть распространён на орграфы.

**Примечание:** Алгоритм Флёри считается не эффективным, поскольку имеет асимптотическую временную сложность  $O(|E|^2)$ .

Реализация: С одним из вариантов реализации алгоритма Флёри можно ознакомиться на веб-сервис для хостинга и совместной разработке IT-проектов GitHub по ссылке <https://github.com/dkulig/fleury-algorithm>.

## Примеры решения ряда электроэнергетических задач с использованием теории графов

- Примечание:** Электрическая сеть – техническая система, предназначенная для передачи и распределения электрической энергии с поддержанием требуемого уровня надёжности (непрерывности) энергетических потоков и допустимости его параметров (прежде всего напряжения).
- Примечание:** Режимное состояние (режим) – состояние технической системы, характеризующееся (не)полнотой выполнения функций, конфигурацией внутренней структуры и значениями параметров в точках контроля (в общем случае во всех элементах системы).
- Примечание:** Коммутационное состояние электрической сети является компонентом её режимного состояния.
- Примечание:** Анализ коммутационного состояния позволяет качественно оценить выполнение основных функций (транспорта электроэнергии, надёжности транспорта, поддержания живучести, поддержания ремонтпригодности, экономичности транспорта, качества доставляемой электроэнергии) и идентифицировать класс состояния электрической сети (например, оптимальное, допустимое с отклонениями от оптимального, аварийное).
- Примечание:** Маршрутизация энергетических потоков (перекоммутации) адаптирует коммутационное состояние электрической сети к характеру этих потоков, меняет полноту (не)выполнения функций, устраняет аварийные последствия спорадических нарушений в электрической сети, позволяет выводить оборудование для проведения ремонтных работ и регулирования напряжения.

Существует ряд электроэнергетических задач, для решения которых применяется теория графов:

- анализ коммутационного состояния электрической сети,
- перевод электрической сети из одного класса состояния в другой,
- синтез электрической сети, удовлетворяющей целевым (функциональным, структурным) показателям
- и другие.

В задаче анализа коммутационного состояния электрической сети выделяются процедуры:

- определения подсети,
- построения иерархической модели сети,
- построения агрегированной модели сети,
- оценки полноты выполнения сетью своих функций
- и другие.

Далее приведен пример решения одной из электроэнергетических задач анализа коммутационного состояния электрической сети с использованием теории графов – определение подсети.

## Определение подсети. Пример решения

**Примечание:** Определение базовых понятий как правило конструируется аксиоматическим методом – заданием ряда атрибутивных свойств и их значений. Из них логическим путём (посредством доказательств) выводятся все остальные понятия.

**Примечание:** Определение можно понимать и как задание некоторой границы «распространения» объекта, свойства, его значения. Она может быть выражена через граничные элементы, характерные структурные свойства или область (минимальную, максимальную) выполнения ряда функций.

**Примечание:** Переход от понятия к категории подразумевает формализацию функции членения-синтезирования.

**Примечание:** Понятие электрической сети определяется из её назначения. Применение к нему фреймового многоуровневого рассмотрения (представления) объектов искусственного происхождения позволяет представить сеть в виде совокупности вложенных (под)сетей. Каждый уровень сети характеризуется определёнными требованиями к множеству выполняемых функций и необходимым для их выполнения набором элементов, коммутации между которыми позволяют реконфигурировать сеть и, соответственно, влиять на полноту выполнения функций.

Известны трудности неавтоматического заполнения баз данных, начиная от низкой производительности и качества ручного исполнения этой работы и заканчивая её экономической нецелесообразностью.

Один из путей преодоления указанных трудностей – автоматическое распознавание информации, содержащейся в чертежах коммутационных схем электрических сетей.

Последующая аналитическая обработка введённой схемы позволяет определять без участия оператора такие свойства элементов как:

- свойство выключателя быть шиносоединительным, секционным или обходным,
- свойство разъединителя быть линейным, трансформаторным или шинным,
- свойство шины быть обходной и др.

Так же схема коммутации электрической содержит необходимую и достаточную информацию для автоматического поиска границ функциональных подсистем:

- ячейка,
- распределительное устройство,
- подстанция,
- район электрических сетей.

Далее предлагается рассмотреть решение этой задачи с применением категории функционального уровня сети, определённой через атрибутивные элементы и процесс членения.

### Математическая постановка задачи

Дано: Граф электрической сети, соответствующий её однолинейной схеме замещения в виде  $G(V, E)$ , где  $V$  – множество вершин;  $E$  – множество рёбер.

Множество прототипов функциональных уровней сети  $S^p \{S_1^p, S_2^p, \dots, S_k^p, \dots, S_{n-1}^p, S_n^p\}$ ,

где  $n$  – число классов уровней,  $S_k^p(V_k^p, E_k^p)$  – прототип  $k$ -го функционального уровня сети.

Здесь  $V_k^p = \emptyset$ ,  $E_k^p$  – множество классов продольных рёбер, ограничивающих уровень, образуемый от данного прототипа.

#### Примечание:

В данной постановке задачи элементы электрической сети представляются рёбрами.

Возможны, и используются, в зависимости от особенностей решаемых задач, другие представления электрической сети в виде графа, например:

- вершины – некоммутируемые элементы электрической сети (линии, шины, трансформаторы), рёбра – коммутационные аппараты (выключатели, разъединители и другие);
- вершины – источники (электрические станции), потребители (подстанции, распределительные пункты), рёбра – линии электропередачи.

Найти: Все подграфы вида  $S_k^L(V_k^L, E_k^L)$  графа  $G(V, E)$ .  $E_k^L$  – содержит подмножества:

$E_{k,1}^L$  – внутренних рёбер (элементов) графа уровня,  $E_{k,2}^L$  – ограничивающих уровень рёбер (элементов).

Решение: Предлагается решение задачи методом поясного наращивания (поиск в ширину) от  $e \in E_1$  – ребра, неограничивающего определяемый уровень, к  $e \{ \dots \} \subset E_2$  – ограничивающим рёбрам.

Для  $\forall S_k^p$  и  $\forall e \in E_1$ :  $E_{j,i+1}^{L,e} = E_{j,i}^{L,e} + D_{j,i}^{L,e}$ , где  $j = 1, 2$  – номер подмножества рёбер,  $i$

– номер шага поясного наращивания,  $E_{j,i}^{L,e}$ ,  $E_{j,i+1}^{L,e}$  – множество  $E_j^{L,e}$  на  $i$ -ом и  $i+1$ -ом шагах,

$D_{j,i}^{L,e}$  – множество (подмножество  $E$ ) связей, образующих пояс вокруг  $E_{j,i}^{L,e}$ , на  $i$ -ом шаге.

Условие завершения построения подграфа  $S_k^p$ , образованного от  $G(V, E)$ :  $D_{j,i}^{L,e} = \emptyset$ .

При построении подграфа  $S_k^p$  от следующего  $e \in E_1$  могут быть введены дополнительные условия, например,

$$\begin{cases} \forall e^f \notin E_1^{L,d}; \\ f \neq d; \\ f, d = 1 \dots m. \end{cases}$$

**Примечание:** Это условие справедливо для схем, содержащих распределительные устройства с целым числом выключателей на цепь и не секционированных по числу цепей (исключая схемы типа «многоугольник»).

Тогда перед формированием  $S_k^{L,e_2}$  необходимо произвести вычитание  $E = E - E_1^{L,e_1}$ .

Описание алгоритма решения задачи:

Алгоритм автоматического решения задачи определения функциональных уровней сети основан на применении функции с циклической обработкой данных.

Аргументами (входными переменными) являются:

- $G(V, E)$  – граф электрической сети,
- $S^P$  – множество прототипов функциональных уровней сети.

Результатом является  $S^L$  – множество функциональных уровней сети (подграфов графа  $G(V, E)$ ).

Условие завершения – разметка принадлежности всех элементов на всех уровнях сети.

Обобщённая блок-схема алгоритма автоматического и ручного определения подсетей представлена на рисунках 5-8.

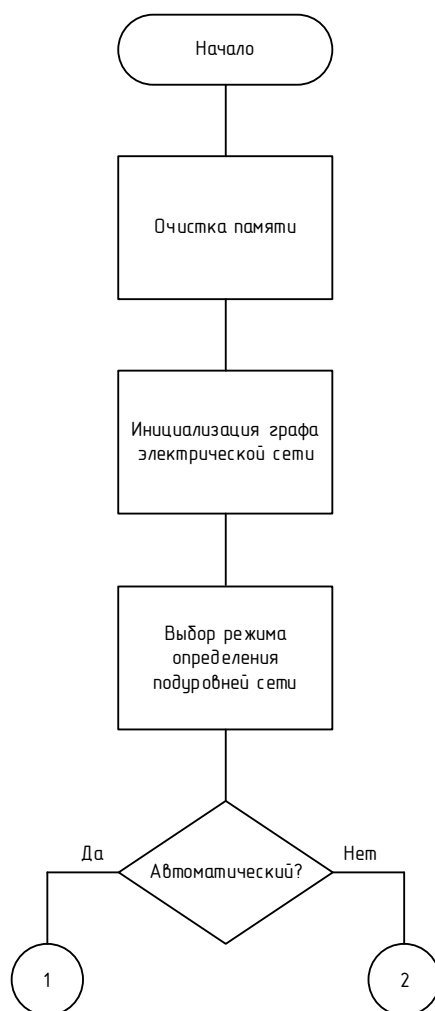


Рисунок 5 – Обобщённая блок-схема алгоритма определения подсетей

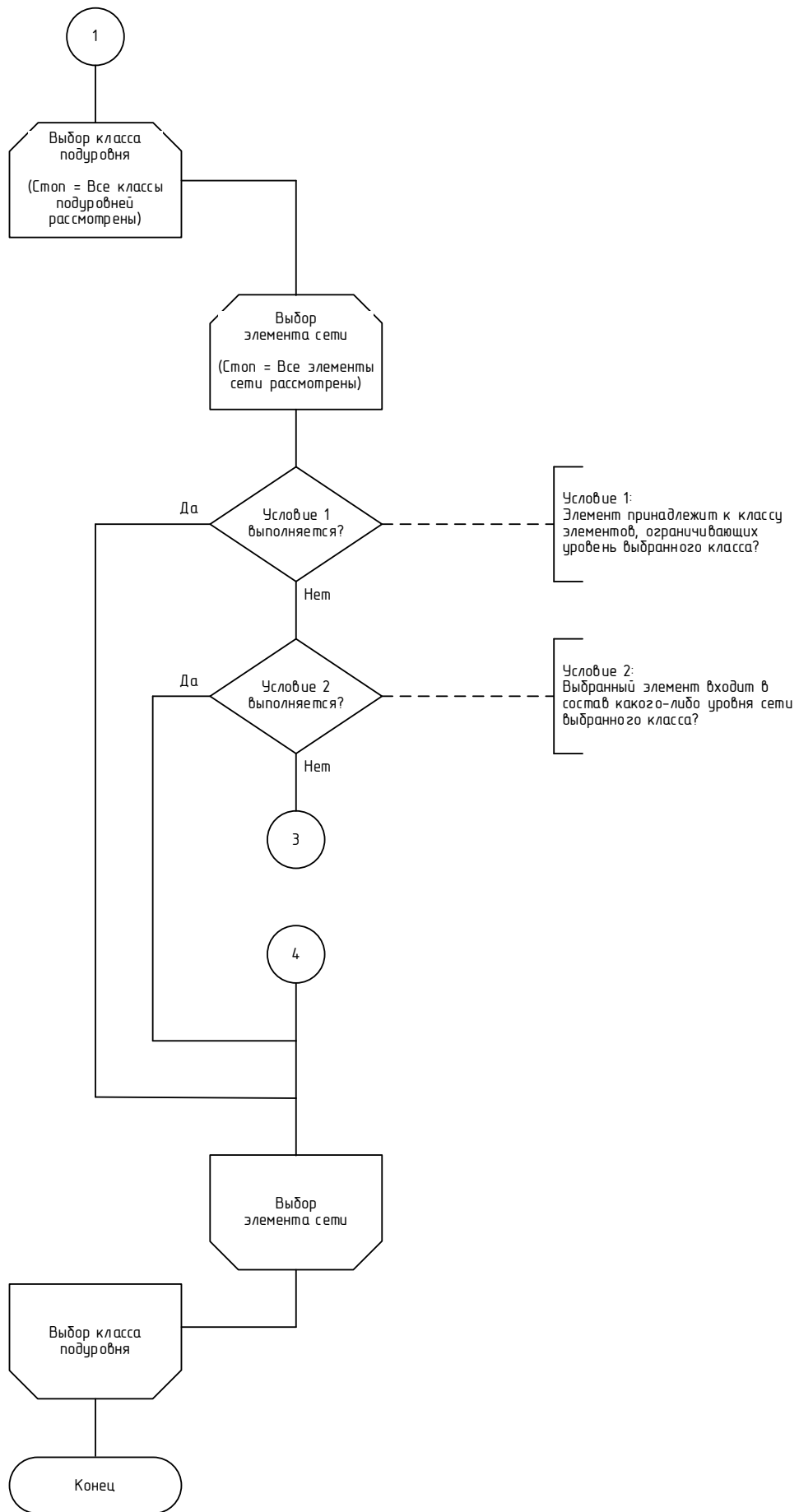




Рисунок 6 – Обобщённая блок-схема алгоритма определения подсетей (продолжение)

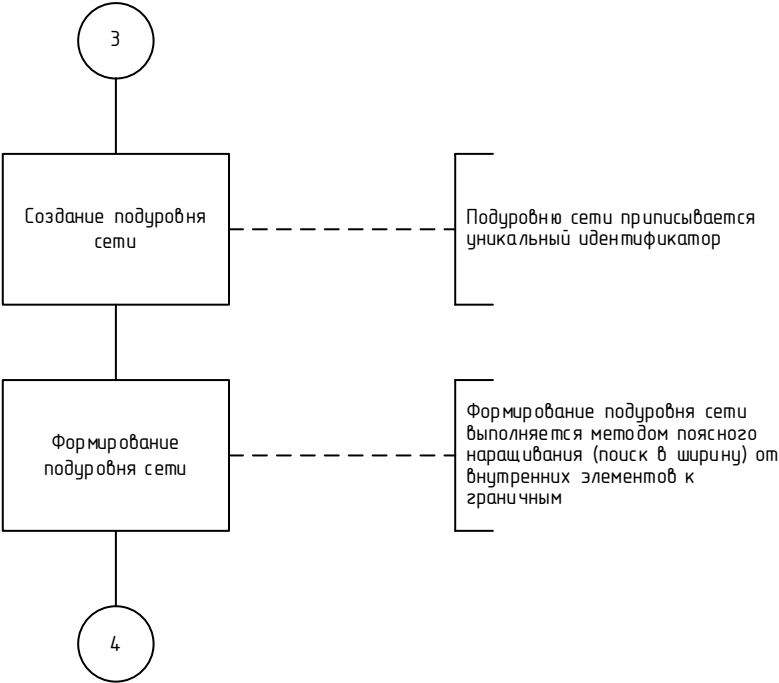


Рисунок 7 – Обобщённая блок-схема алгоритма определения подсетей (продолжение)

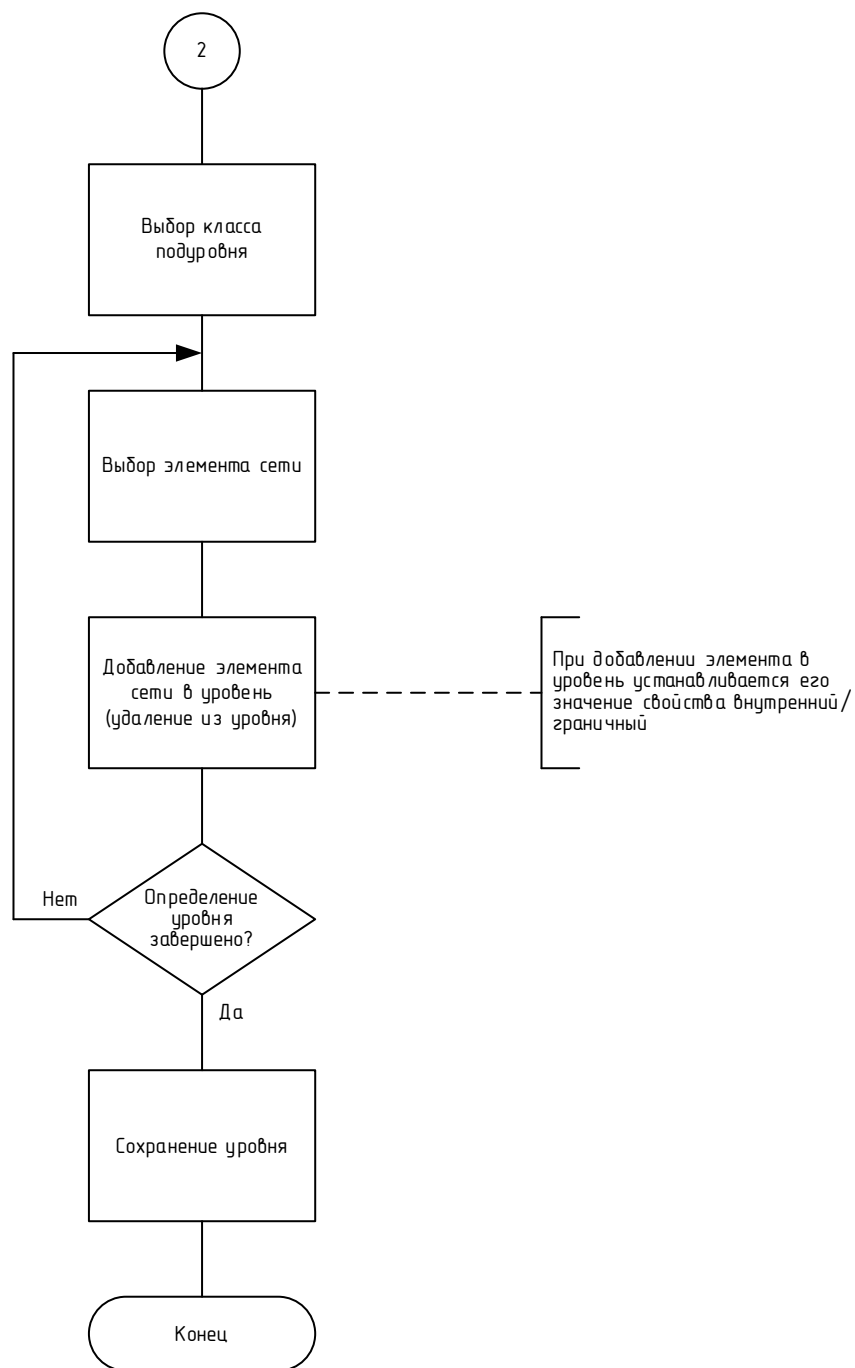


Рисунок 8 – Обобщённая блок-схема алгоритма определения подсетей (завершение)

### Пример решения задачи

В качестве *элементарного* примера покажем решение задачи определения одной из ячеек подстанции «Ночка» (Новосибирскэнерго).

Дано:

1. Схема подстанции показана на рисунке 6.
2. Граф однолинейной схемы замещения подстанции, заданный списком инцидентности в таблице 5.
3. Прототип подсети "ячейка" задан множеством классов продольных рёбер, ограничивающих уровень, в таблице 6.

Найти: Граф ячейки, в которую входит разъединитель ЛР ВН-1.

Решение: Представим решение в форме таблицы 7. Рёбра графа, принадлежащие множеству  $E_2$

будем выделять серым цветом, принадлежность рёбер к  $E_1$  особо отмечать не будем. Условием принадлежности ребра к множеству  $E_2$  является принадлежность класса элемента (рёбра)  $C_e$  к множеству классов элементов, задающих прототип подуровня «ячейка».

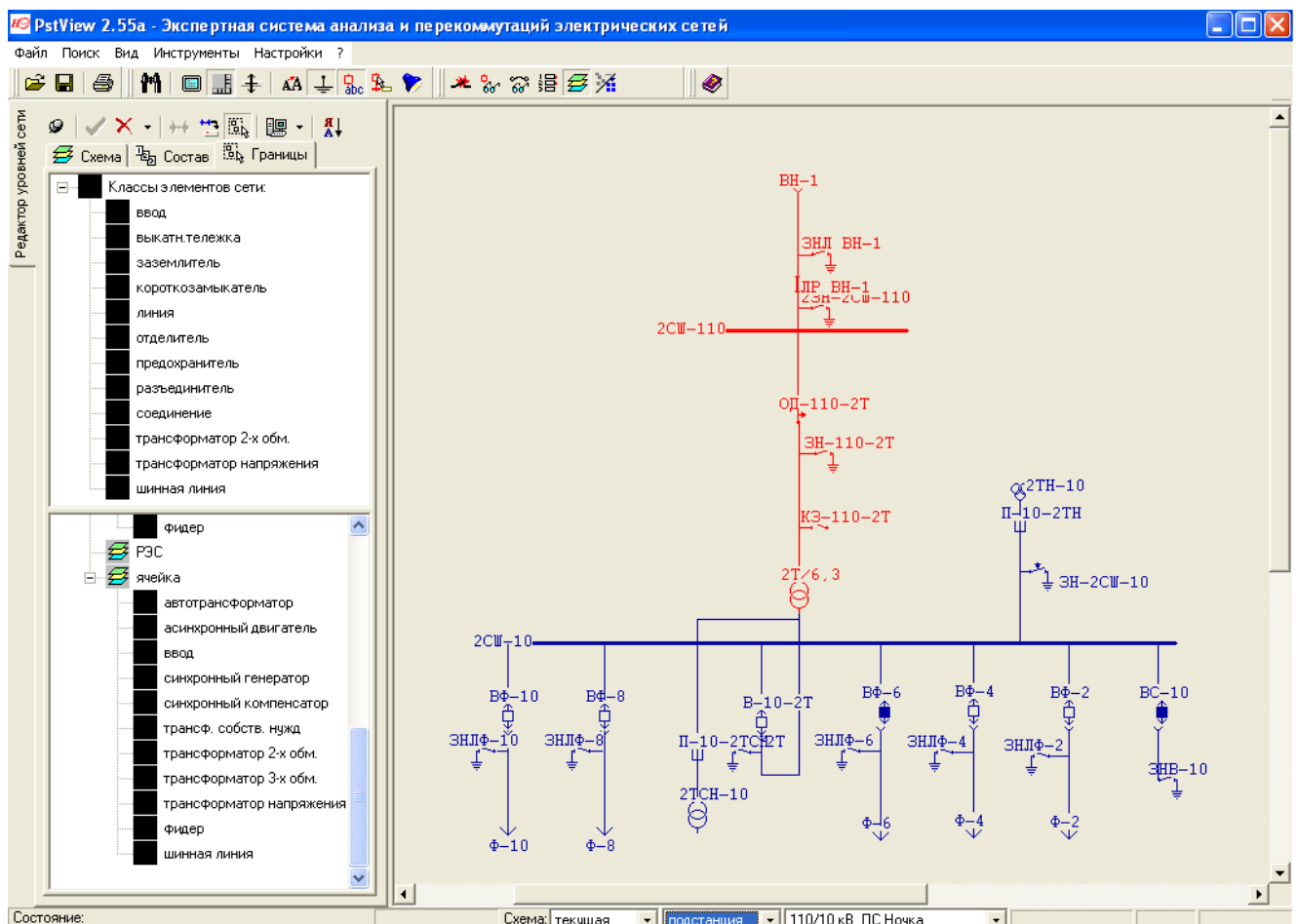


Рисунок 6 – Схема подстанции «Ночка» (Новосибирскэнерго).

Таблица 5 – Граф однолинейной схемы замещения ПС Ночка (Новосибирскэнерго)

№ пп	Начало $V_e$	Конец $W_e$	Класс элемента $C_e$	Наименование класса элемента	Наименование элемента
1	2	3	4	5	6
1	7	4	19	разъединитель	ЛР ВН-1
2	8	9	1	соединение	-
3	10	8	14	трансформатор 2-х обм.	2Т/6,3
4	12	0	3	шинная линия	2СШ-10
5	12	11	1	соединение	-
6	14	13	1	соединение	-
7	13	15	1	соединение	-
8	9	15	1	соединение	-
9	16	14	1	соединение	-
10	11	14	9	выкатн.тележка	В-10-2Т
11	12	17	1	соединение	-
12	19	18	1	соединение	-
13	20	19	1	соединение	-
14	18	0	37	фидер	Ф-10
15	21	0	8	заземлитель	ЗНВ-10
16	23	22	1	соединение	-
17	12	24	1	соединение	-
18	22	25	1	соединение	-
19	27	26	1	соединение	-
20	29	28	1	соединение	-
21	30	21	1	соединение	-
22	31	0	37	линия	Ф-6
23	25	31	1	соединение	-
24	26	32	1	соединение	-
25	28	33	1	соединение	-
26	32	0	37	линия	Ф-4
27	33	0	37	линия	Ф-2
28	23	0	8	заземлитель	ЗНЛФ-6
29	34	0	8	заземлитель	ЗНЛФ-4
30	35	0	8	заземлитель	ЗНЛФ-2
31	12	36	1	соединение	-
32	12	37	1	соединение	-
33	12	38	1	соединение	-
34	24	22	9	выкатн.тележка	ВФ-6
35	37	27	9	выкатн.тележка	ВФ-4
36	38	29	9	выкатн.тележка	ВФ-2
37	36	30	9	выкатн.тележка	ВС-10
38	34	27	1	соединение	-
39	35	29	1	соединение	-
40	39	40	9	выкатн.тележка	ВФ-8
41	12	39	1	соединение	-
42	40	41	1	соединение	-

Продолжение таблицы 5

43	42	40	1	соединение	-
44	41	0	37	фидер	Ф-8
45	42	0	8	заземлитель	ЗНЛФ-8
46	20	0	8	заземлитель	ЗНЛФ-10
47	43	44	1	соединение	-
48	17	19	9	выкатн.тележка	ВФ-10
49	45	12	1	соединение	-
50	47	46	1	соединение	-
51	48	45	1	соединение	-
52	46	48	5	предохранитель	П-10-2ТН
53	45	49	1	соединение	-
54	49	0	8	заземлитель	ЗН-2СШ-10
55	47	0	28	трансформатор напряжения	2ТН-10
56	16	0	8	заземлитель	ЗН-10-2Т
57	50	8	1	соединение	-
58	50	51	1	соединение	-
59	51	43	5	предохранитель	П-10-2ТСН
60	44	0	14	трансформатор 2-х обм.	2ТСН-10
61	52	0	8	заземлитель	ЗН-110-2Т
62	5	53	1	соединение	-
63	53	54	20	отделитель	ОД-110-2Т
64	54	10	1	соединение	-
65	54	55	1	соединение	-
66	55	0	4	короткозамыкатель	КЗ-110-2Т
67	54	52	1	соединение	-
68	5	0	3	шинная линия	2СШ-110
69	1	0	38	ввод	ВН-1
70	2	0	8	заземлитель	ЗНЛ ВН-1
71	3	0	8	заземлитель	2ЗН-2СШ-110
72	4	5	1	соединение	-
73	4	3	1	соединение	-
74	6	2	1	соединение	-
75	6	7	1	соединение	-
76	1	6	1	соединение	-

Таблица 6 – Прототип подсети «ячейка»

№ пп	Код класса элементов	Наименование класса элементов
1	2	3
1	3	Шинная линия
2	11	Синхронный генератор
3	12	Синхронный компенсатор
4	13	Асинхронный двигатель
5	14	Трансформатор 2-х обмоточный
6	15	Трансформатор 3-х обмоточный
7	16	АТ
8	17	ТСН
9	28	ТН
10	37	Фидер
11	38	Ввод

Таблица 7 – Решение задачи определения подсети "ячейка" для разъединителя ЛР ВН-1

№ шага	Граф ячейки на i-ом шаге $E_i^{\text{яч. ЛР ВН-1}} \{ \{v_e, w_e, c_e\}, \dots \}$	Множество рёбер пояса наращивания на i-ом шаге $D_i^{\text{яч. ЛР ВН-1}} \{ \{v_e, w_e, c_e\}, \dots \}$	Примечание
1	2	3	4
1	$\{7, 4, 19\}$	$\{ \{4, 5, 1\}, \{4, 3, 1\}, \{6, 7, 1\} \}$	Начальное ребро – разъединитель ЛР ВН-1
2	$\{ \{7, 4, 19\}, \{4, 5, 1\}, \{4, 3, 1\}, \{6, 7, 1\} \}$	$\{ \{1, 6, 1\}, \{5, 0, 3\}, \{3, 0, 8\}, \{6, 2, 1\} \}$	Ребро $\{5, 0, 3\}$ – шинная линия, принадлежит подмножеству $E_{2,i}^{\text{яч. ЛР ВН-1}}$ . Поиск других рёбер $v_e = 5$ или $w_e = 5$ не производится
3	$\{ \{7, 4, 19\}, \{4, 5, 1\}, \{4, 3, 1\}, \{6, 7, 1\}, \{1, 6, 1\}, \{5, 0, 3\}, \{3, 0, 8\}, \{6, 2, 1\} \}$	$\{ \{1, 0, 38\}, \{2, 0, 8\} \}$	Ребро $\{1, 0, 38\}$ – шинная линия, принадлежит подмножеству $E_{2,i}^{\text{яч. ЛР ВН-1}}$ . Поиск других рёбер $v_e = 1$ или $w_e = 1$ не производится
4	$\{ \{7, 4, 19\}, \{4, 5, 1\}, \{4, 3, 1\}, \{6, 7, 1\}, \{1, 6, 1\}, \{5, 0, 3\}, \{3, 0, 8\}, \{6, 2, 1\}, \{1, 0, 38\}, \{2, 0, 8\} \}$	$\emptyset$	Завершение

### Контрольные задания

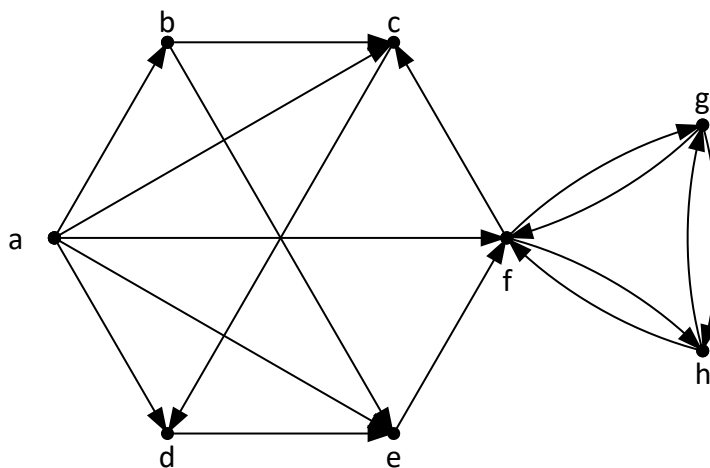
1. Какие способы представления графов Вам известны?
2. Какие способы представления слабосвязанных графов считаются наиболее «экономичными» по отношению к занимаемой памяти?
3. Каким способ представлен граф G?

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

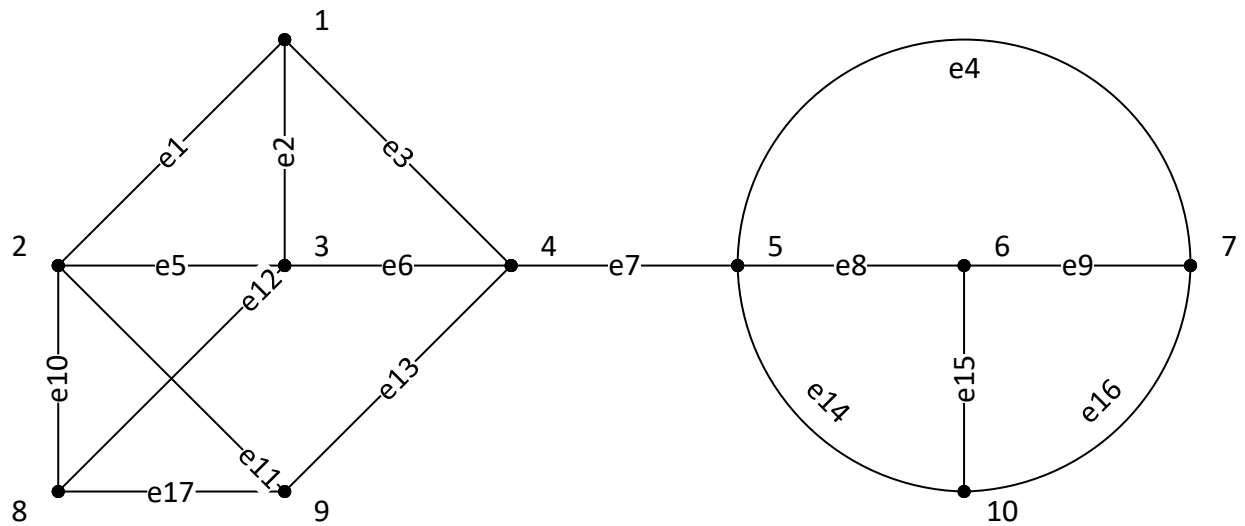
4. Каким способом представлен граф G в коде на языке Python?

```
#объявление графа G
a, b, c, d, e, f, g, h = range(8)
N = [
    {b, c, d, e, f}, # a
    {c, e}, # b
    {d}, # c
    {e}, # d
    {f}, # e
    {c, g, h}, # f
    {f, h}, # g
    {f, g} # h
]
```

5. Перечислите известные Вам основные числовые характеристики графа G и их значения.



6. Перечислите известные Вам основные числовые характеристики графа G и их значения.



7. Представьте граф G (см. п. 6) в виде списка инцидентности и запишите последовательность обхода графа в ширину, начиная с вершины 1.

8. Представьте граф G (см. п. 6) в виде списка инцидентности и запишите последовательность обхода графа в глубину, начиная с вершины 1.

9. Перечислите примеры электроэнергетических задач, для решения которых может быть использована теория графов.

10. Поясните синтаксис и изложите действие метода `Queue.get(block=True, timeout=None)`, содержащегося в классе `Queue` модуля `queue` языка Python.

11. Поясните синтаксис и изложите действие метода `Queue.put(item, block=True, timeout=None)`, содержащегося в классе `Queue` модуля `queue` языка Python.

12. Поясните синтаксис и изложите действие метода `Queue.Empty()`, содержащегося в классе `Queue` модуля `queue` языка Python.



## Приложение 1. Варианты заданий

Варианты заданий даны в таблице П1.

Таблица П1 – Варианты заданий

Вариант	Дано	Найти
1	2	3
1	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – список инцидентности.	1. Степени всех вершин. 2. Число компонентов связности. 3. Преобразовать граф к виду – матрица смежности. 4. Степени всех чётных вершин (п. 3)
2	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – матрица инцидентности.	1. Степени всех чётных вершин. 2. Число компонентов связности. 3. Преобразовать граф к виду – список смежности. 4. Степени всех вершин (п. 3)
3	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – матрица смежности.	1. Степени всех нечётных вершин. 2. Число компонентов связности. 3. Преобразовать граф к виду – список инцидентности. 4. Степени всех вершин (п. 3)
4	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – список смежности.	1. Степени всех вершин. 2. Число компонентов связности. 3. Преобразовать граф к виду – матрица инцидентности. 4. Степени всех вершин (п. 3)
5	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – список инцидентности.	1. Степени всех вершин. 2. Число компонентов связности. 3. Построить граф, в котором вершины соответствуют рёбрам исходного графа, а рёбра - вершинам. 4. Степени всех чётных вершин (п. 3)
6	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – матрица инцидентности.	1. Степени всех чётных вершин. 2. Число компонентов связности. 3. Построить граф, в котором вершины соответствуют рёбрам исходного графа, а рёбра - вершинам. 4. Степени всех вершин (п. 3)
7	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – матрица смежности.	1. Степени всех нечётных вершин. 2. Число компонентов связности. 3. Построить граф, в котором вершины соответствуют рёбрам исходного графа, а рёбра - вершинам. 4. Степени всех вершин (п. 3)
8	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – список смежности.	1. Степени всех вершин. 2. Число компонентов связности. 3. Построить граф, в котором вершины соответствуют рёбрам исходного графа, а рёбра - вершинам. 4. Степени всех вершин (п. 3)

Продолжение таблицы П1

9	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – список инцидентности.	1. Степени всех вершин. 2. Число компонентов связности. 3. Число контуров
10	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – матрица инцидентности.	1. Степени всех вершин. 2. Число компонентов связности. 3. Число контуров
11	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – матрица смежности.	1. Степени всех вершин. 2. Число компонентов связности. 3. Число контуров
12	Случайный ненаправленный граф (число вершин не более 12, число рёбер не более 24). Способ задания графа – список смежности.	1. Степени всех вершин. 2. Число компонентов связности. 3. Число контуров