# Лекция 3
# Математические модели и методы теории графов, используемые в САПР КЭС

# Вопросы лекции

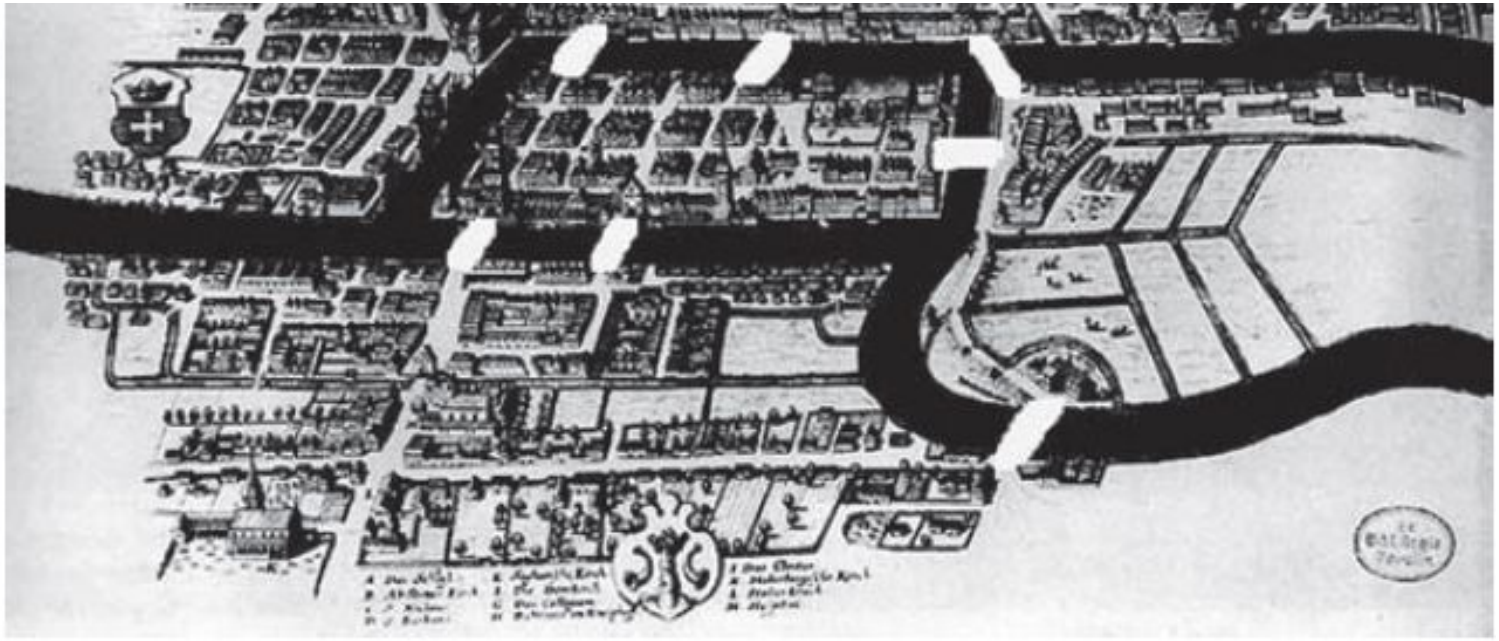
1. Основные понятия теории графов.
2. Задачи анализа и синтеза электронных средств и систем, решаемые методами теории графов.
3. Методы поиска кратчайших маршрутов на графах.

# Вопрос 1.
# Основные понятия теории графов

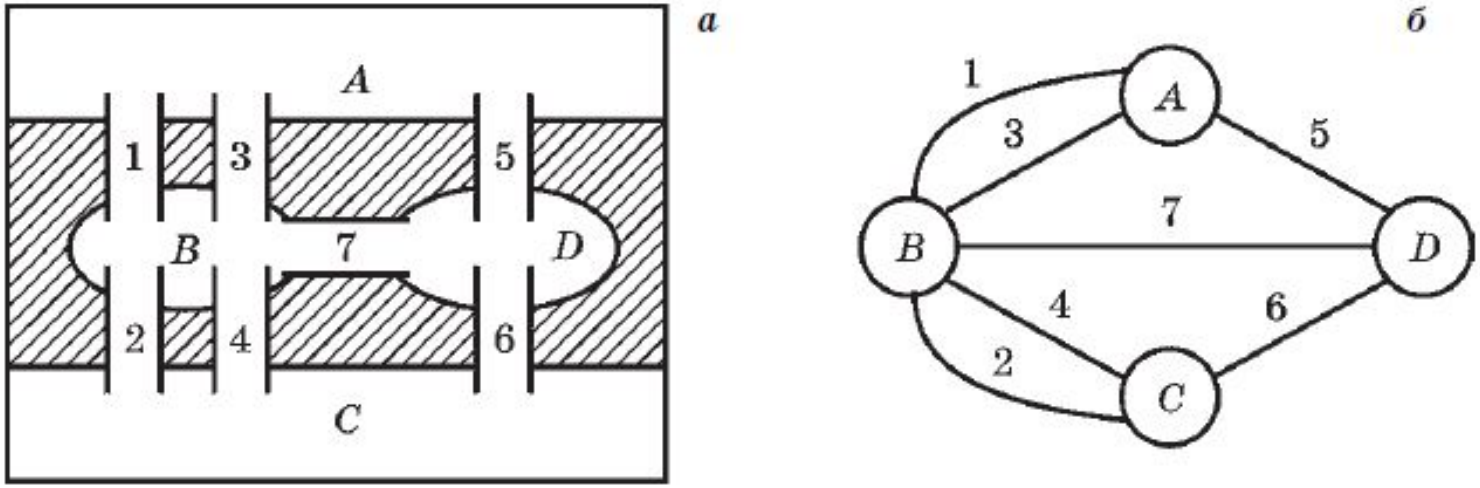
Теорией графов называется раздел дискретной математики, изучающий свойства графов. В общем смысле граф представляется как множество вершин (узлов), соединенных ребрами (связями). С математической точки зрения граф представляет собой совокупность двух множеств — вершин и ребер, т. е. $G(V, R)$; $V=\{v_i\}$, $R=\{r_k\}$.

Строго говоря, $V$ является подмножеством любого счетного множества, а $R$ — подмножеством $V \times V$.

Родоначальником теории графов считается немецкий и русский математик, член Петербургской академии наук Леонард Эйлер. В 1736 г. в одном из своих писем он формулирует и предлагает решение задачи о семи кенигсбергских мостах, ставшей впоследствии одной из классических задач теории графов.



Мосты парка города Кенигсберга в 1736 г.




*а* — упрощенная схема мостов; *б* — граф мостов.

*Граф и способы его задания*

**Графом** $G[V, U]$ будем называть совокупность множества вершин (узлов) $\{v_k\} = V$, $k = \overline{1, n}$; и множества ветвей $\{u_{i,j}\} = U$, $i = \overline{1, n}$, $j = \overline{1, n}$, соединяющих между собой все вершины или часть их.

Элементы множества $V$ называют по-разному: вершины, узлы, события, точки и т. д.

Элементы множества $U$ также имеют различные названия: ветви, дуги, ребра, транзиты, работы.

Ненаправленная ветвь $u_{i,j}$ называется **ребром**. В этом случае порядок следования вершин не учитывается.

Направленная ветвь $u_{i,j}$ называется **дугой**. В данном случае порядок следования вершин учитывается: дуги $u_{i,j}$ и $u_{j,i}$ считаются различными.

Граф, состоящий из вершин и ребер, называется неориентированным.

Граф, состоящий из вершин и дуг, называется ориентированным, или **орграфом**.

Среди большого числа способов задания графов наиболее часто используют геометрический (графический) и матричный (табличный).

При геометрическом способе задания графов на плоскости точкой или кружком изображаются вершины графа, а соединяющей их линией — ребро или дуга графа. На графе сначала нумеруют вершины, а затем проставляют индексы ветвей. На рис. 2.1 и 2.2 изображены неориентированный и ориентированный графы.

Существуют также смешанные графы, которые содержат как ребра, так и дуги.

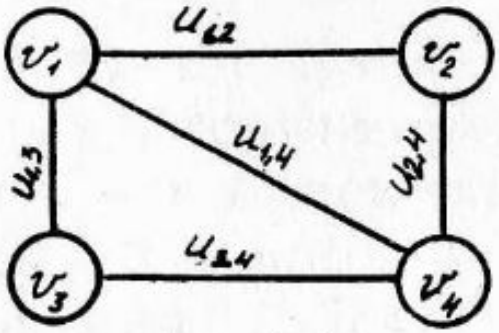



Рис. 2.1

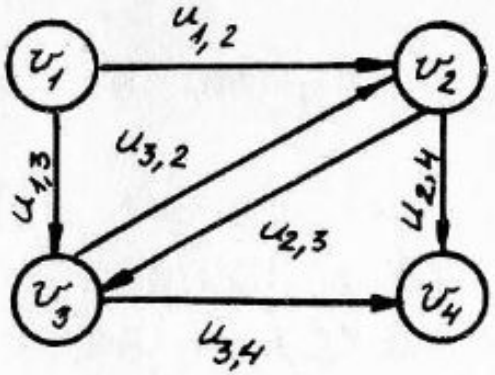



Рис. 2.2

Матричный способ задания графов более удобен при работе с ЭВМ. В зависимости от существа конкретной решаемой задачи можно применять различные матрицы, однако в основном используют матрицу инциденций и матрицу смежности (соседства).

Матрица инциденций отображает связь между узлами с учетом направления и имеет размерность $n \times m$, где $n$ — число узлов, $m$ — число дуг.

Матрицу инциденций принято обозначать следующим образом:

$$R_I = [r_{i,j}], \quad i = 1, n, \quad j = 1, m.$$

Число строк равно числу узлов, а число столбцов — числу дуг. Элементы матрицы инциденций определяются по правилу

$$r_{i,j} = \begin{cases} 1, \text{ если } u_{i,j} \text{ исходит из } v_i; \\ -1, \text{ если } u_{i,j} \text{ входит в } v_i; \\ 0 \text{ в остальных случаях.} \end{cases}$$

В матрице инциденций неориентированного графа все отличные от нуля элементы положительны.

Матрица смежности (соседства) определяет связь узлов друг с другом и имеет размерность $n \times n$, где $n$ — число узлов.

Обозначение матрицы смежности:

$$R_S = [r_{i,j}], \quad i = \overline{1, n}, \quad j = \overline{1, n}.$$

Строки и столбцы соответствуют узлам графа. Элементы матрицы смежности определяются по правилу

$$r_{i,j} = \begin{cases} 1, \text{ если } v_i \text{ соединена с } v_j \text{ дугой } u_{i,j}; \\ 0 \text{ в остальных случаях.} \end{cases}$$

Для графа, который изображен на рис. 2.2, матрица инциденций имеет вид

$$R_I = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccccc} u_{1,2} & u_{1,3} & u_{2,3} & u_{2,4} & u_{3,2} & u_{3,4} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & -1 & 0 \\ 0 & -1 & -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 & -1 \end{pmatrix} \end{array}.$$

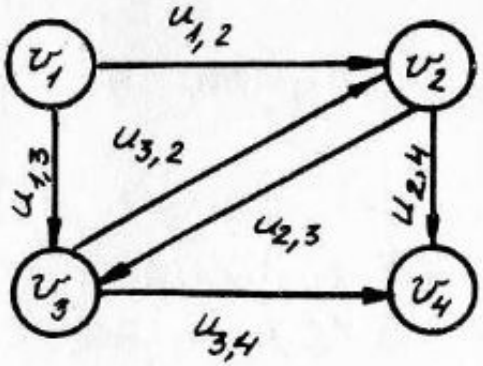



Рис. 2.2

Для графа, который изображен на рис. 2.2, матрица смежности имеет вид

$$R_S = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}.$$

Элементы главной диагонали матрицы $R_S$ всегда равны нулю. Для неориентированных графов матрица смежности симметрична.

Для полного задания графа нужно не только описать его структуру, но и ввести данные, определяющие параметры узлов и дуг (ребер).

В зависимости от конкретного типа физической системы параметры узлов и дуг (ребер) графа могут быть различными. Применительно к сетям связи параметрами узлов могут быть мощность, быстродействие, надежность и т. д., параметрами дуг — пропускная способность, канальная емкость, надежность, длина, стоимость и т. д.

Для организаций, которые занимаются строительством сетей связи, важным параметром является протяженность линий, так как она определяет трудовые и материальные затраты. Для тех, кто эксплуатирует сети связи, более важными могут оказаться другие параметры, например пропускные способности ветвей.

Протяженность ветвей удобно задавать с помощью расстояний. Элементы матрицы расстояний равны:

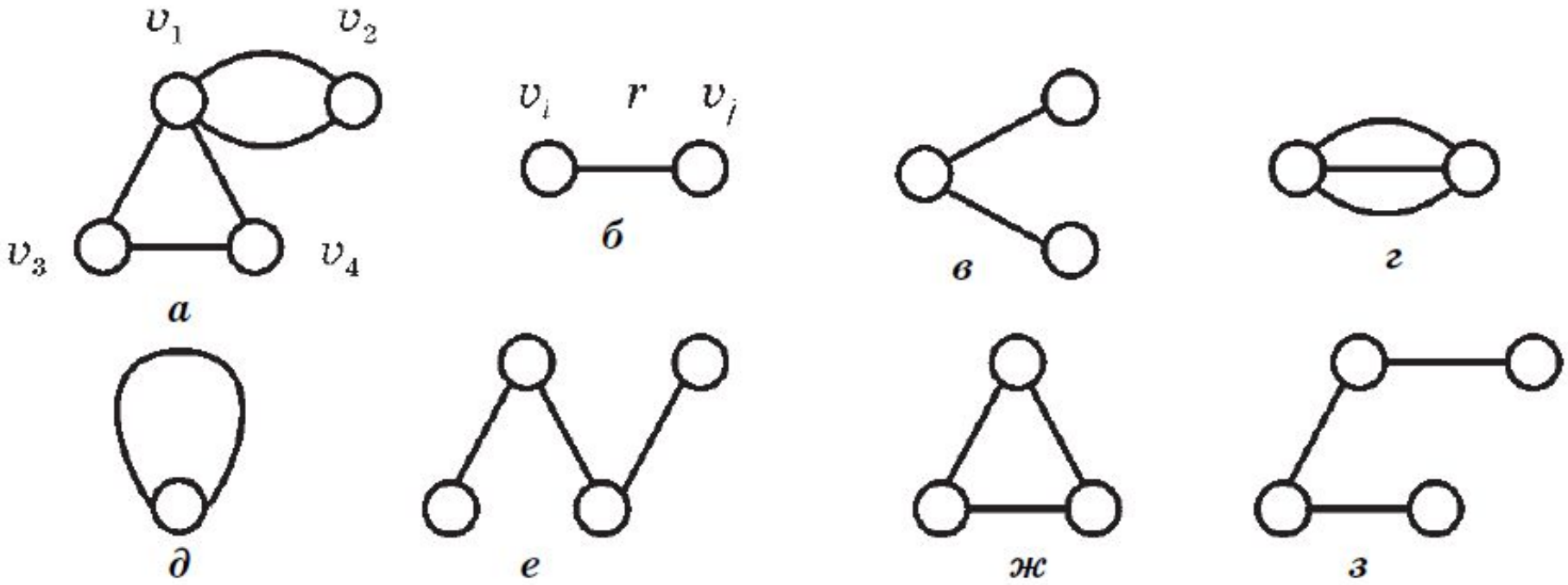
$$r_{i,j} = \begin{cases} 0, \text{ если } i = j; \\ \infty \text{ (или } 0), \text{ если вершины } u_i \text{ и } u_j \text{ не связаны;} \\ L_{i,j} \text{ в остальных случаях.} \end{cases}$$

Здесь $L_{i,j}$ — расстояние между вершинами $u_i$ и $u_j$.

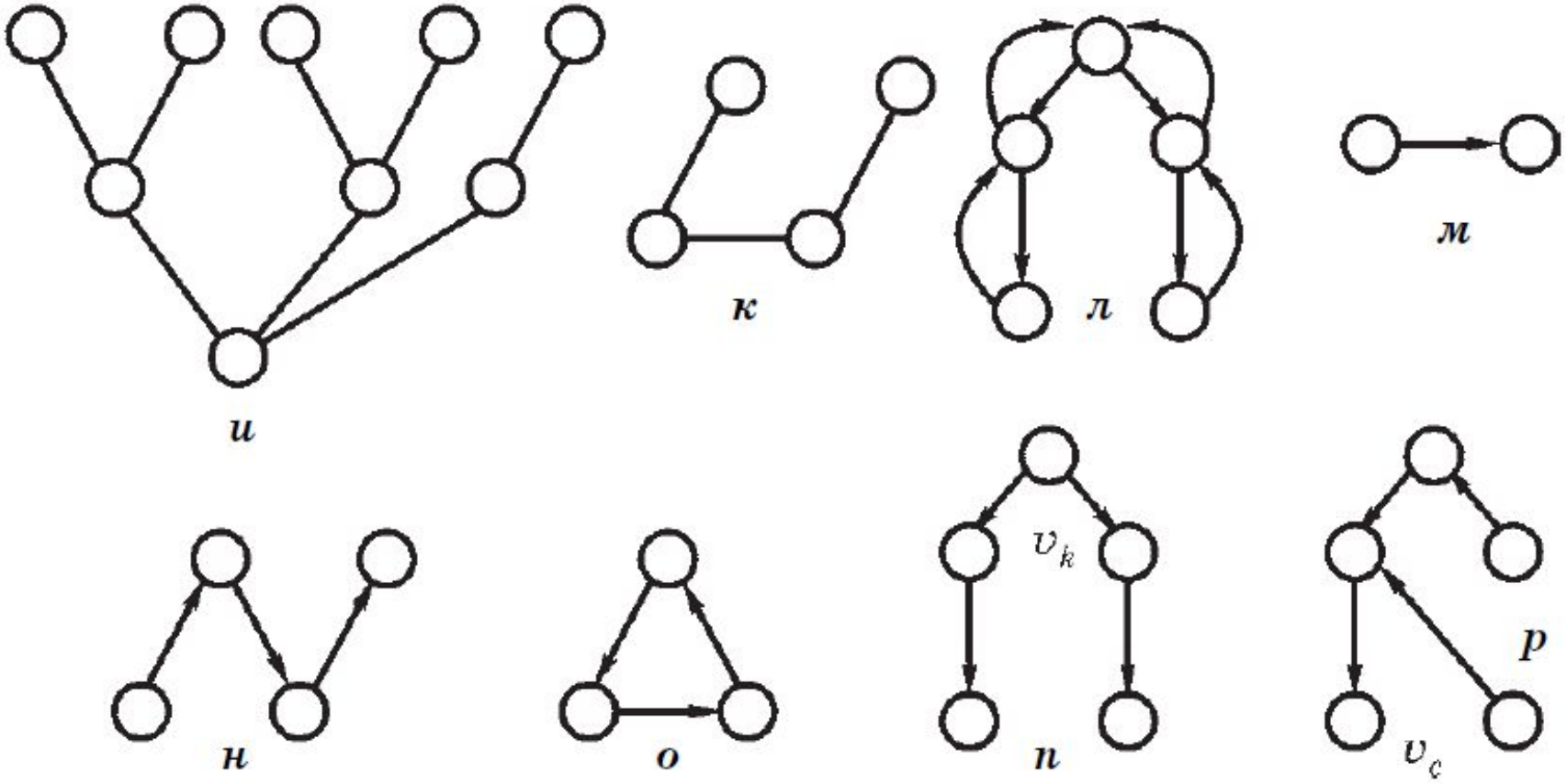
Пропускные способности системы удобно задавать с помощью матрицы пропускных способностей. Элементы матрицы пропускных способностей равны:

$$r_{i,j} = \begin{cases} 0 \text{ (или } \infty), \text{ если } i = j; \\ 0, \text{ если вершины } u_i \text{ и } u_j \text{ не связаны;} \\ C_{i,j} \text{ в остальных случаях.} \end{cases}$$

Здесь $C_{i,j}$ — пропускная способность ребра, например, максимальное число сообщений в единицу времени.




*а* — граф; *б* — ребро с граничными вершинами; *в* — смежные ребра; *г* — кратныеребра; *д* — петля; *е* — цепь; *ж* — цикл; *з* — дерево;




*и* — корневое дерево; *к* — лес; *л* — ориентированный граф;

*м* — дуга с началом $v_i$ и концом $v_j$;

*н* — путь; *о* — контур; *п* — ориентированное дерево с корнем $v_к$;

*р* — остовное входящее дерево со стоком в вершине $v_с$ орграфа *л*.

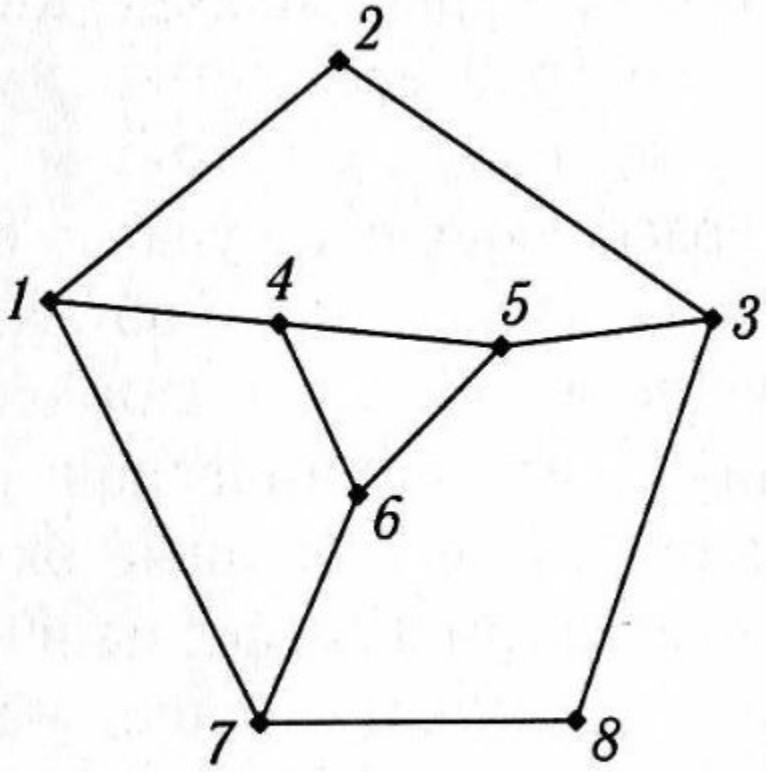



**Рис. 2.** Граф $G$

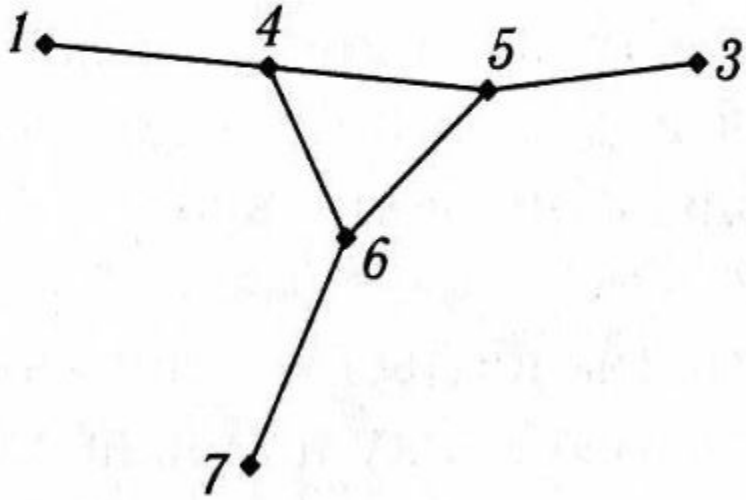



**Рис. 3.** Индуцированный подграф графа $G$ с рис. 2

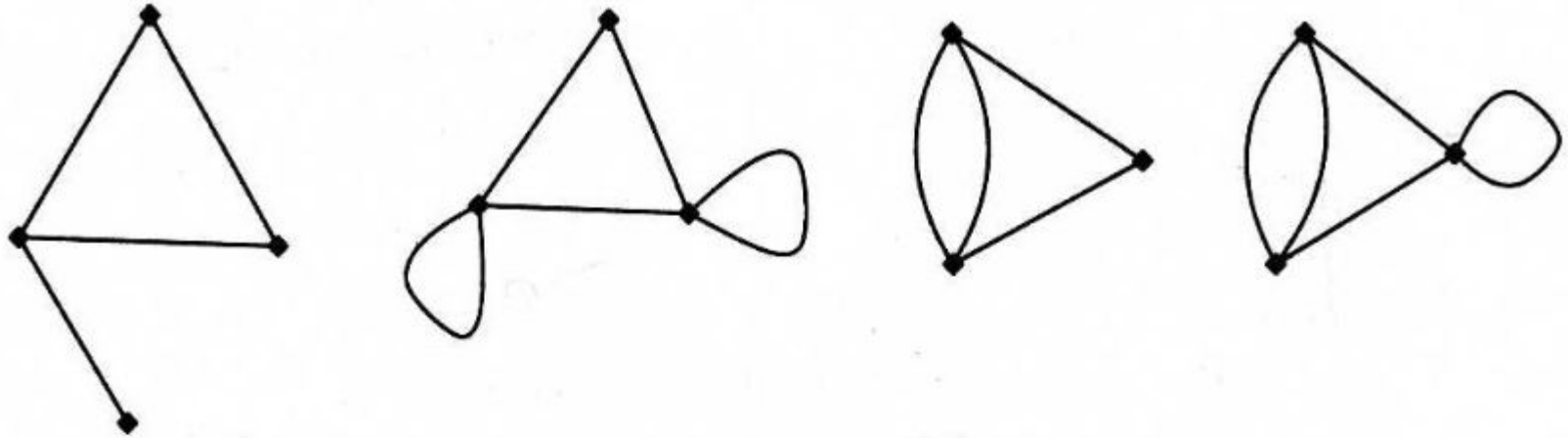


**Рис. 7.** Граф, псевдограф, мультиграф, псевдомультиграф

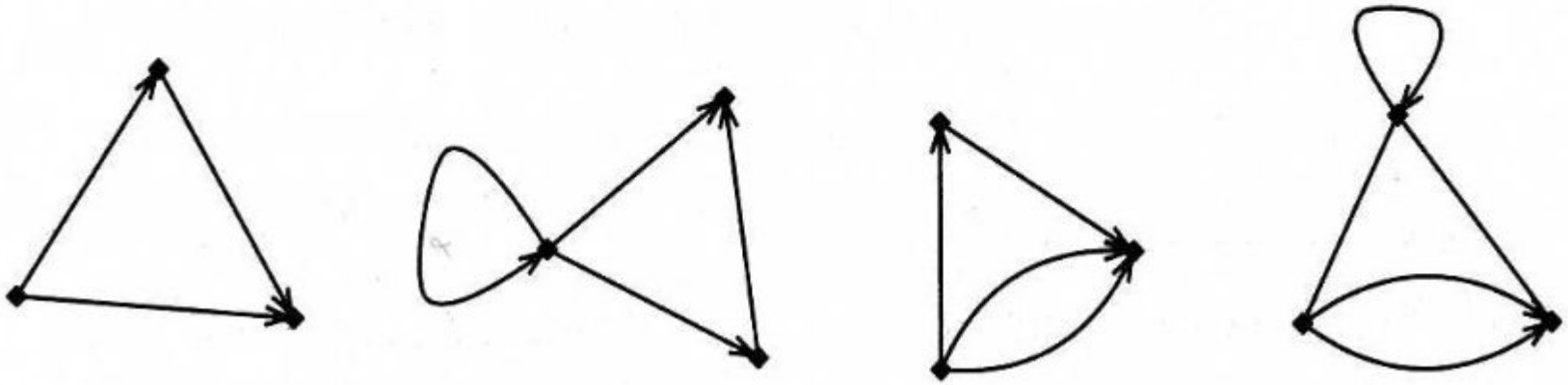


**Рис. 8.** Орграф, псевдоорграф, мультиорграф, псевдомультиорграф

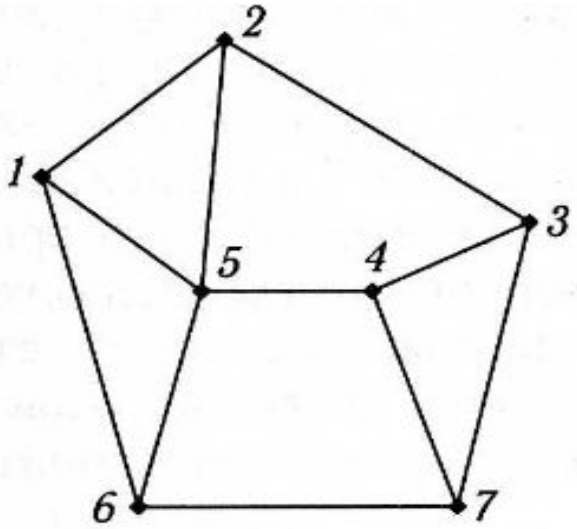



**Рис. 9.** Граф $G$

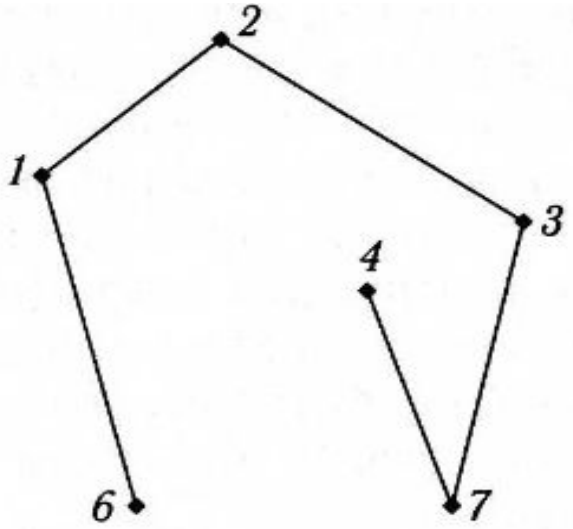



**Рис. 10.** Простая цепь в графе $G$ с рис. 9

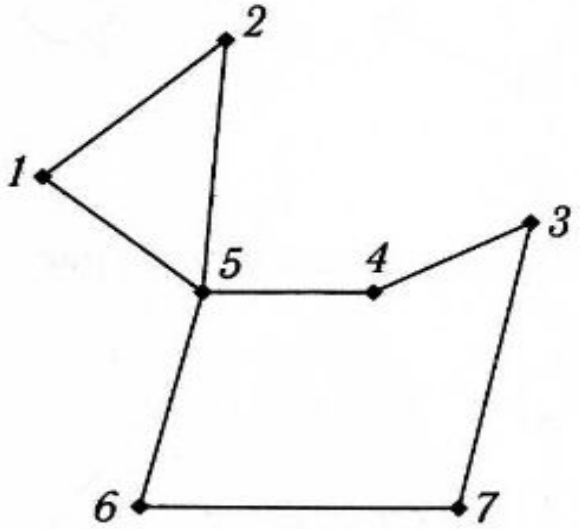



**Рис. 11.** Цикл в графе $G$ с рис. 9

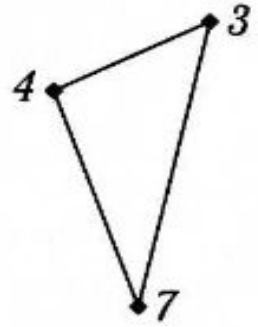



**Рис. 12.** Простой цикл в графе $G$ с рис. 9

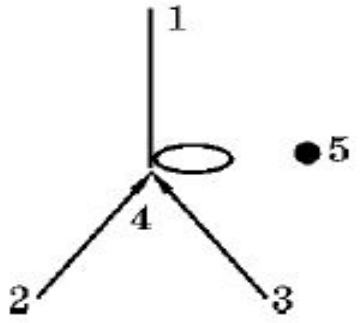
**Определение 1.1.** *Степенью* или *валентностью* вершины $a$ неорграфа $G$ называется число ребер, инцидентных вершине $a$, т. е. число ребер, концом которых является вершина $a$. При этом петли считаются дважды.

Если $G$ — орграф, то степени его вершин определяются как степени вершин в соответствующем неорграфе $F(G)$. Аналогично вводится понятие степени вершины в мультиграфах. Степень вершины можно обозначать символами $\deg_G a$, $\deg a$, $d(a)$.

Вершина степени 0 называется *изолированной*, а степени 1 — *концевой* или *висячей*.

**Пример 1.1.** Вершины графа $G$, изображенного на рис. 29, имеют следующие валентности:

$$\deg 1 = \deg 2 = \deg 3 = 1,\ \deg 4 = 5,\ \deg 5 = 0.$$



Валентности вершин графа

Рассмотрим сумму степеней всех вершин графа. Поскольку каждое ребро входит в эту сумму дважды, справедливо следующее утверждение:

**Теорема 1.1** (лемма о рукопожатиях). Сумма степеней всех вершин графа является четным числом и равна удвоенному числу его ребер.

## Расстояния в графе, диаметр, центр, радиус графа

*Утверждение*. Если для двух вершин существует маршрут, связывающий их, то обязательно найдется минимальный маршрут, соединяющий эти вершины. Обозначим длину этого маршрута через d(v, w).

*Определение*. Величину d(v, w) (конечную или бесконечную) будем называть **расстоянием между вершинами v, w**. Это расстояние удовлетворяет аксиомам метрики:

$d(v, w) \geq 0$, причем $d(v, w) = 0$ тогда и только тогда, когда $v=w$;

$d(v, w) = d(w, v)$;

$d(v, w) \leq d(v, u) + d(u, w)$.

*Определение*. **Диаметром** связного графа называется максимально возможное расстояние между двумя его вершинами.

*Определение*. **Центром** графа называется такая вершина, что максимальное расстояние между ней и любой другой вершиной является наименьшим из всех возможных; это расстояние называется **радиусом** графа.

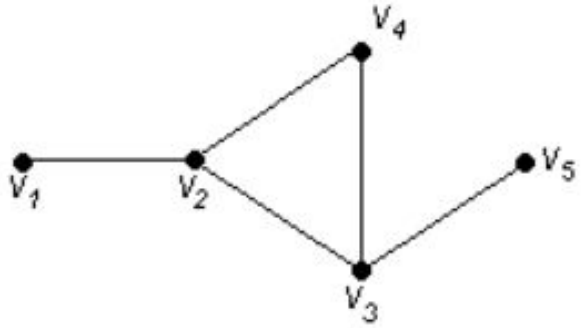



*Рис.* .

Для графа G, изображенного на рисунке , найти радиус, диаметр и центры.

Решение.

Чтобы определить центры, радиус, диаметр графа *G*, найдем матрицу *D(G)* расстояний между вершинами графа, элементами *dij* которой будут расстояния между вершинами *vi* и *vj*. Для этого воспользуемся графическим представлением графа. Заметим, что матрица *D(G)* симметрична относительно главной диагонали.

$$D(G) = \begin{bmatrix} 0 & 1 & 2 & 2 & 3 \\ 1 & 0 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 1 \\ 2 & 1 & 1 & 0 & 2 \\ 3 & 2 & 1 & 2 & 0 \end{bmatrix}.$$

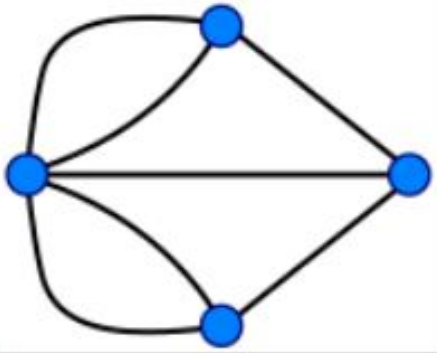
С помощью полученной матрицы для каждой вершины графа *G* определим

$$r(v_i) = \max_j d(v_i, v_j)$$

наибольшее удаление из выражения: для *i, j = 1, 2, ..., 5.*

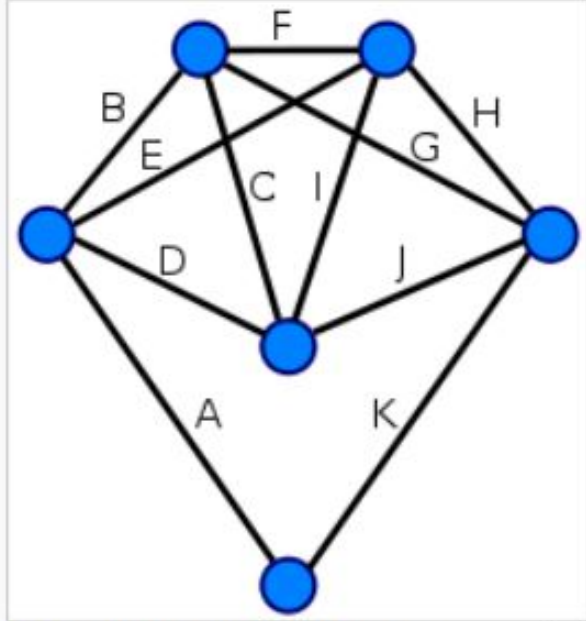
В результате получаем: *r(v1) = 3, r(v2) = 2, r(v3) = 2, r(v4) = 2, r(v5) = 3.* Минимальное из полученных чисел является радиусом графа *G*, максимальное - диаметром графа *G*. Значит, *R(G) = 2* и *D(G) = 3*, центрами являются вершины *v2, v3, v4.*

**Эйлеров путь** (**эйлерова цепь**) в графе — это путь, проходящий по всем рёбрам графа и притом только по одному разу.
**Эйлеров цикл** — эйлеров путь, являющийся циклом. То есть замкнутый путь, проходящий через каждое ребро графа ровно по одному разу.
**Эйлеров граф** — граф, содержащий эйлеров цикл.
**Полуэйлеров граф** — граф, содержащий эйлеров путь
Согласно теореме, доказанной Эйлером (для неориентированного графа), **эйлеров цикл существует** тогда и только тогда, когда граф связный и в нём отсутствуют вершины нечётной степени.



Граф Кёнигсбергских мостов. Этот граф не является эйлеровым, поэтому решения не существует.

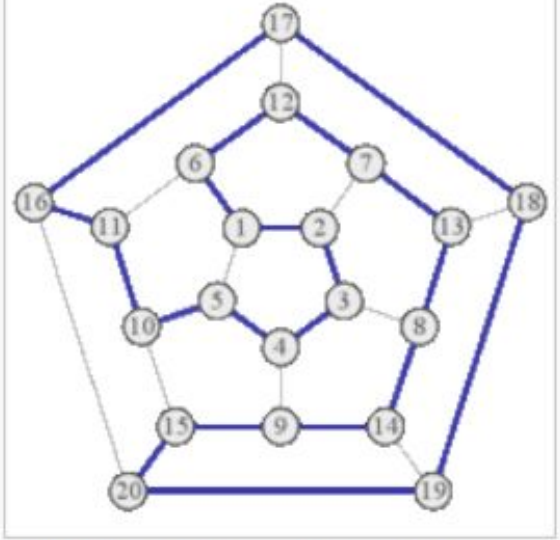
**Эйлеров путь существует** тогда и только тогда, когда граф связный и содержит не более двух вершин нечётной степени. Ввиду леммы о рукопожатиях, число вершин с нечётной степенью должно быть четным. А значит эйлеров путь существует только тогда, когда это число равно нулю или двум. Причём когда оно равно нулю, эйлеров путь вырождается в эйлеров цикл




Каждая вершина этого графа имеет чётную степень, поэтому этот граф — эйлеров. Обход рёбер в алфавитном порядке даёт эйлеров цикл.

**Гамильтонов граф** представляет собой граф, который содержит *гамильтонов цикл*. При этом *гамильтоновым циклом* является такой цикл (замкнутый путь), который содержит все вершины (точки) данного графа.

**Гамильтонов путь** является простым путём (путём без петель), проходящим через каждую вершину графа ровно один раз. Гамильтонов путь отличается от цикла тем, что у пути начальные и конечные точки могут не совпадать, в отличие от цикла. Гамильтонов цикл является гамильтоновым путём.




Гамильтонова линия для додекаэдра, предложенная Гамильтоном для замены его игры «вокруг света» на додекаэдре на задачу для плоского графа.

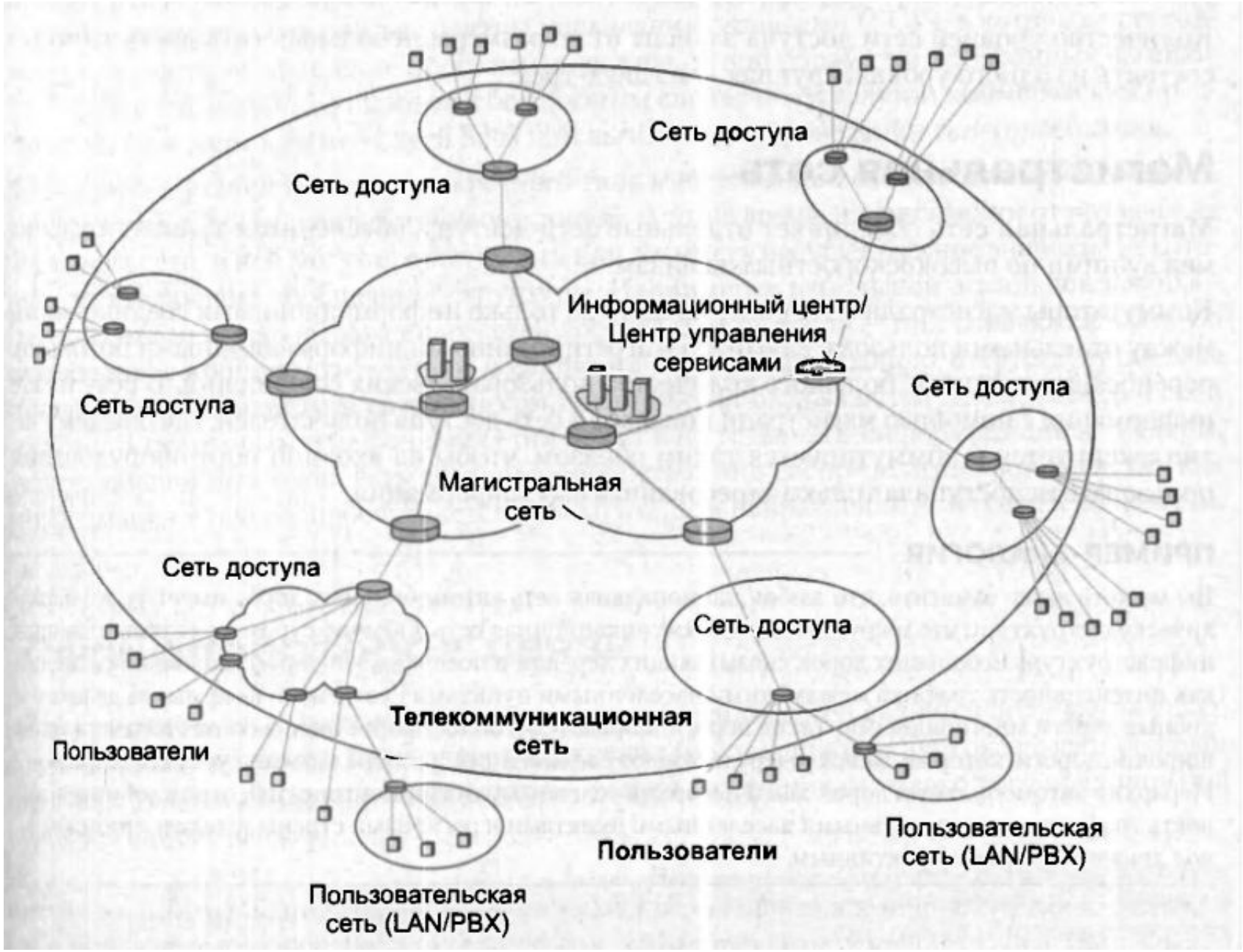
Гамильтоновы путь, цикл и граф названы в честь ирландского математика У.Гамильтона, который впервые определил эти классы, исследовав задачу (игру) **«кругосветного путешествия» по додекаэдру**. В этой задаче вершины додекаэдра символизировали известные города, такие как Брюссель, Амстердам, Эдинбург, Пекин, Прага, Дели, Франкфурт и др., а рёбра — соединяющие их дороги. Путешествующий должен пройти «вокруг света», найдя путь, который проходит через все вершины ровно один раз. Гамильтон предложил вариант игры, заменив додекаэдр плоским графом, изоморфным графу, построенному на рёбрах додекаэдра.

В отличии от эйлеровых графов, где имеется критерий для графа быть эйлеровым, для гамильтоновых графов такого критерия нет, а задача проверки существования гамильтонова цикла оказывается NP-полной. Большинство известных фактов имеет вид: «если граф G имеет достаточное количество ребер, то граф является гамильтоновым».

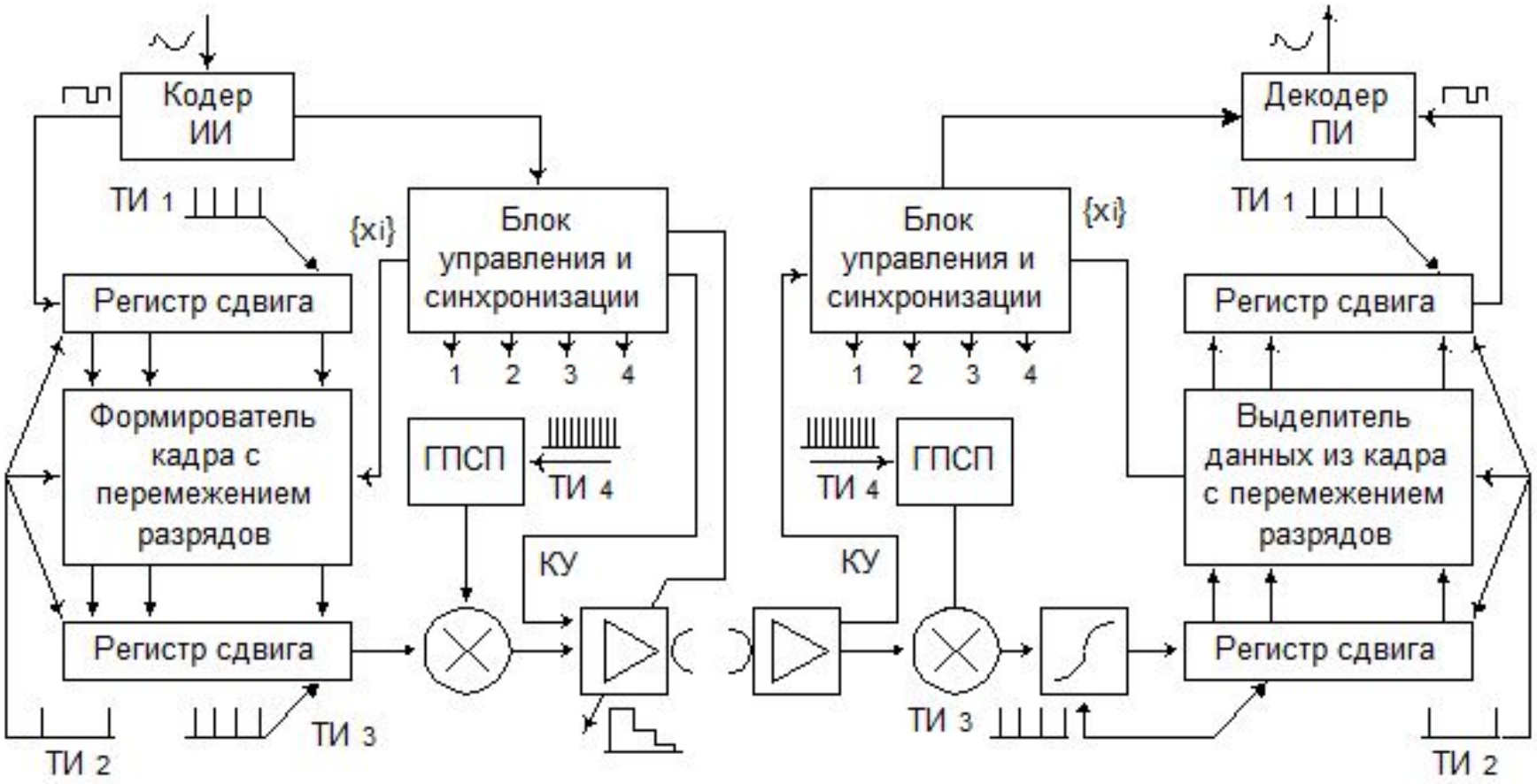
# Вопрос 2

## Задачи анализа и синтеза электронных средств и систем, решаемые методами теории графов

# Граф - схема сети связи



# Граф - функциональная схема электронного средства связи



# Граф - принципиальная схема электронного средства

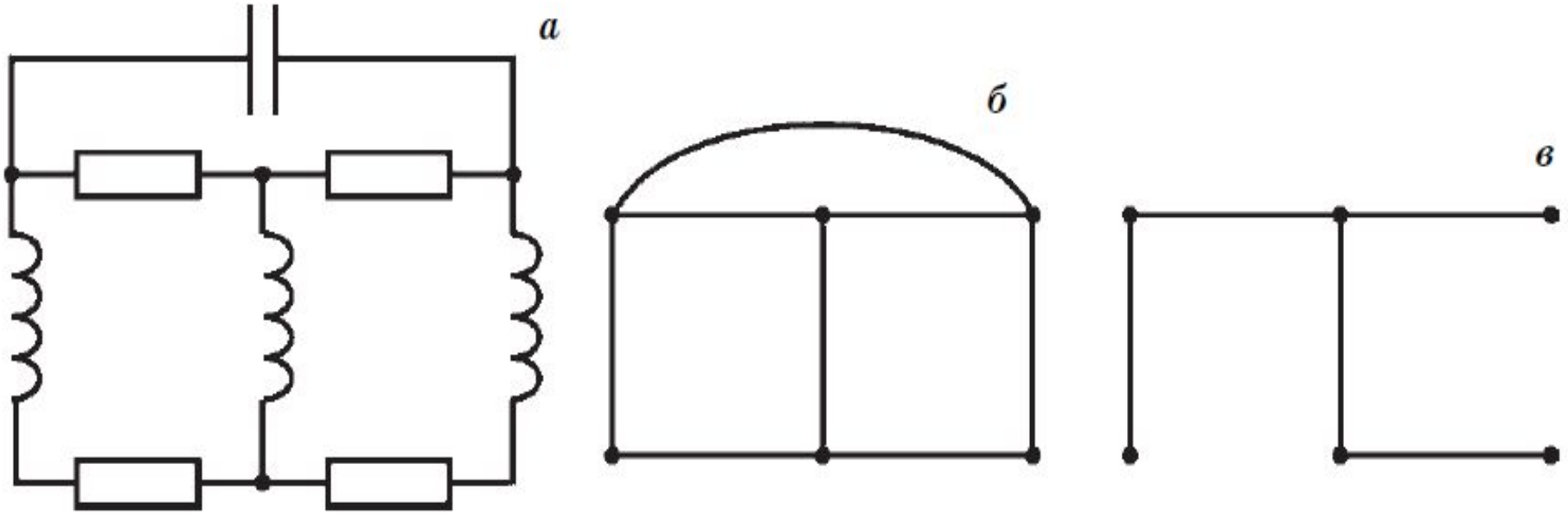



**Рис. 2.5**

Представление электрической цепи в виде графа:

*а* — электрическая цепь; *б* — соответствующий ей граф; *в* — остовное дерево.

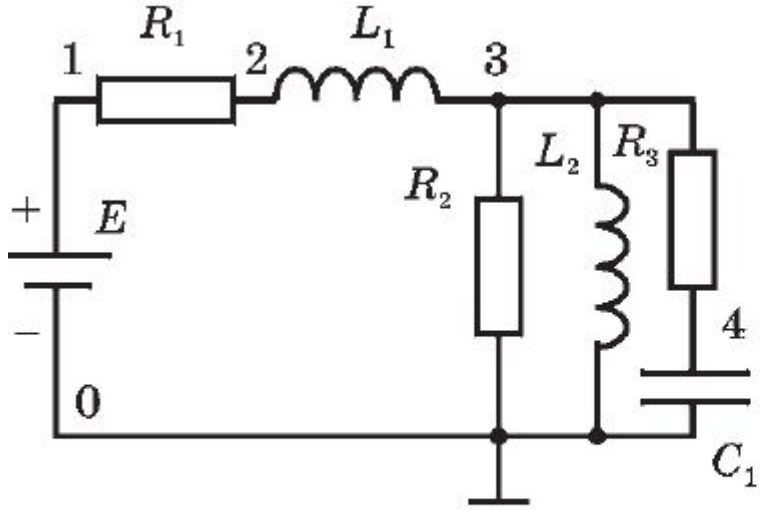



**Рис. 2.6**
Пример электрической схемы

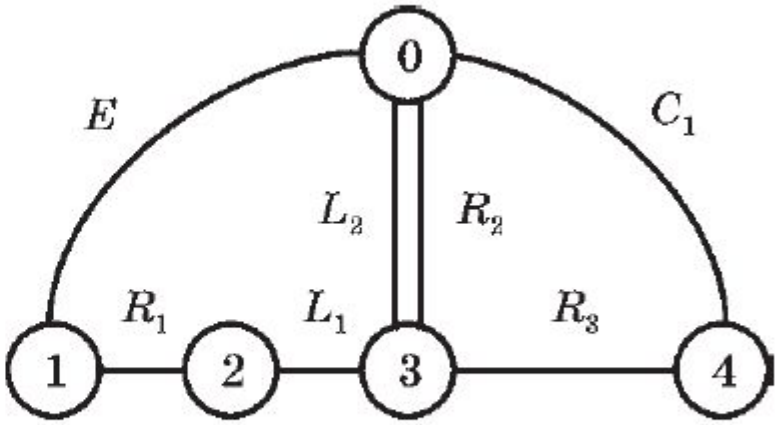



**Рис. 2.7**
Неориентированный граф

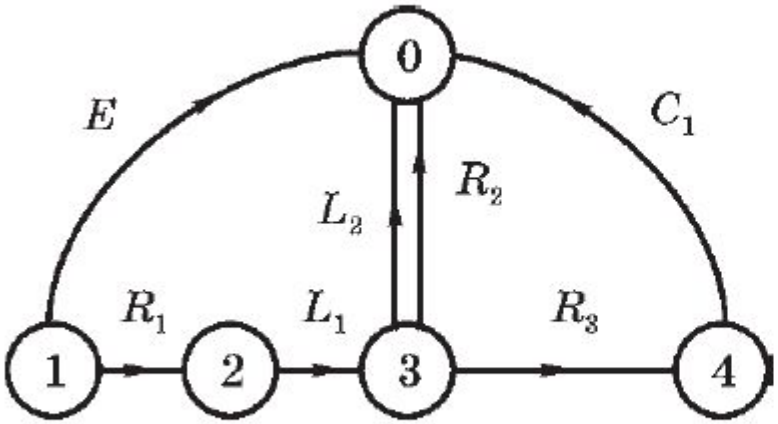



**Рис. 2.8**
Ориентированный граф

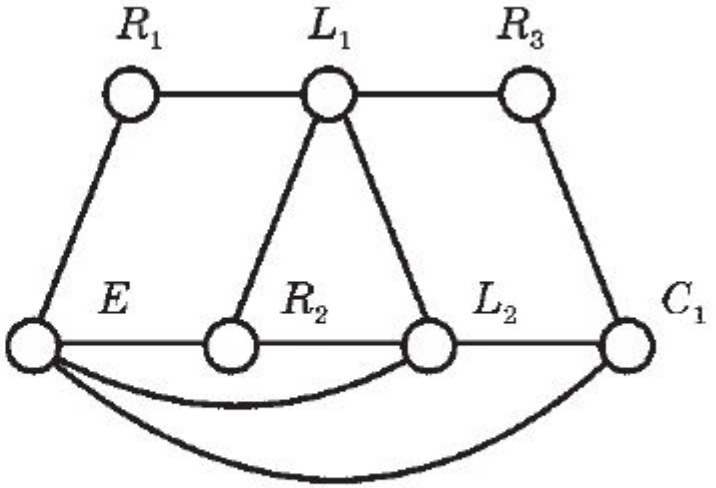



**Рис. 2.9**
Граф связи между элементами

# Задача компоновки

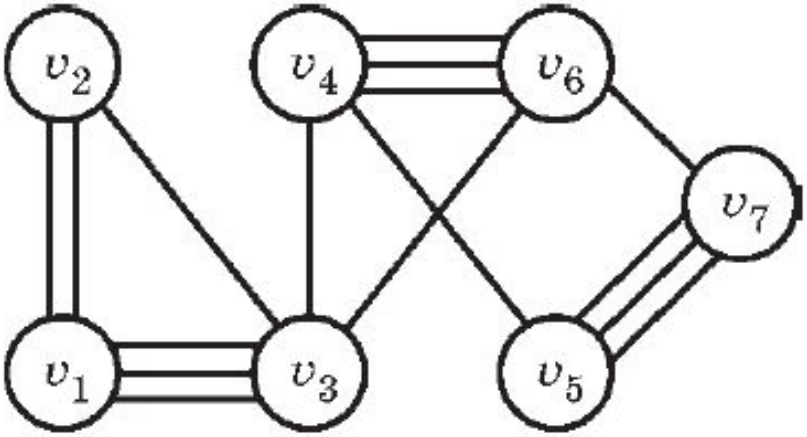



**Рис. 5.1**
Исходный мультиграф

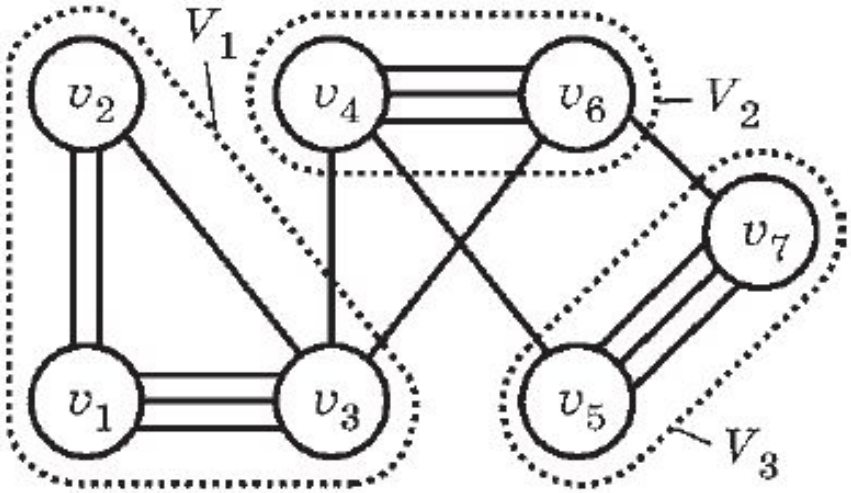



**Рис. 5.2**
Скомпонованный мультиграф

# Задача размещения

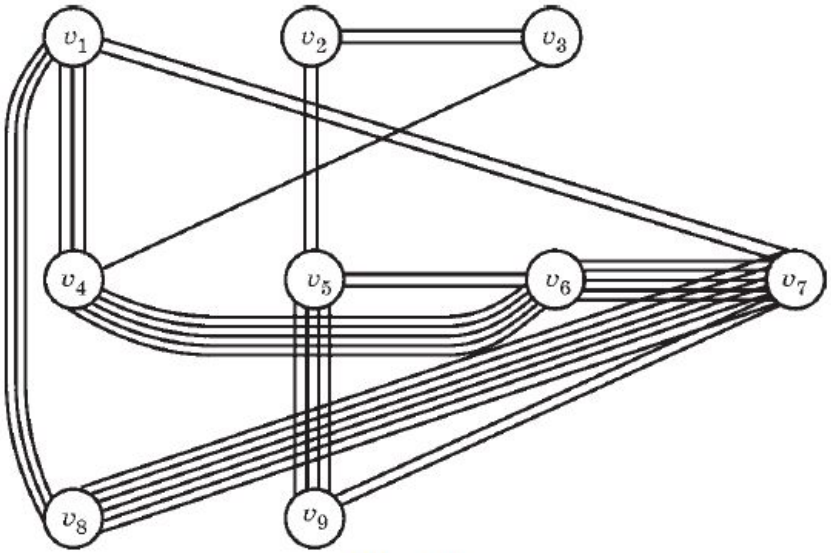



**Рис. 5.11**
Исходный мультиграф

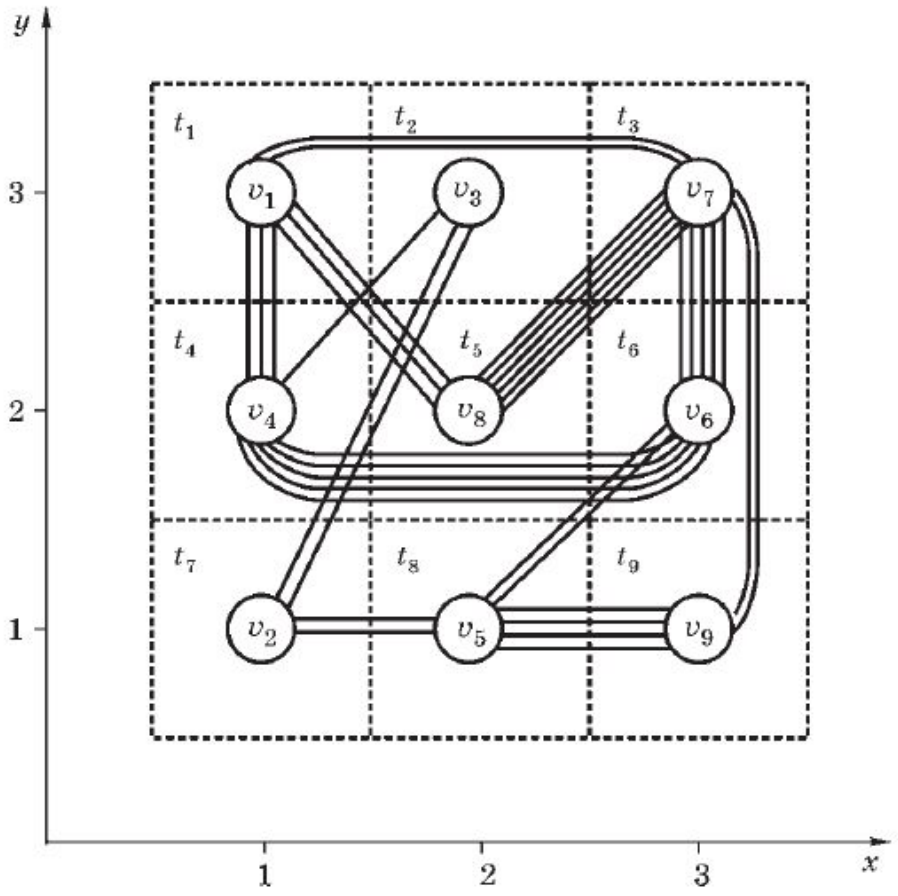



**Рис. 5.13**
Размещение элементов после последовательного этапа

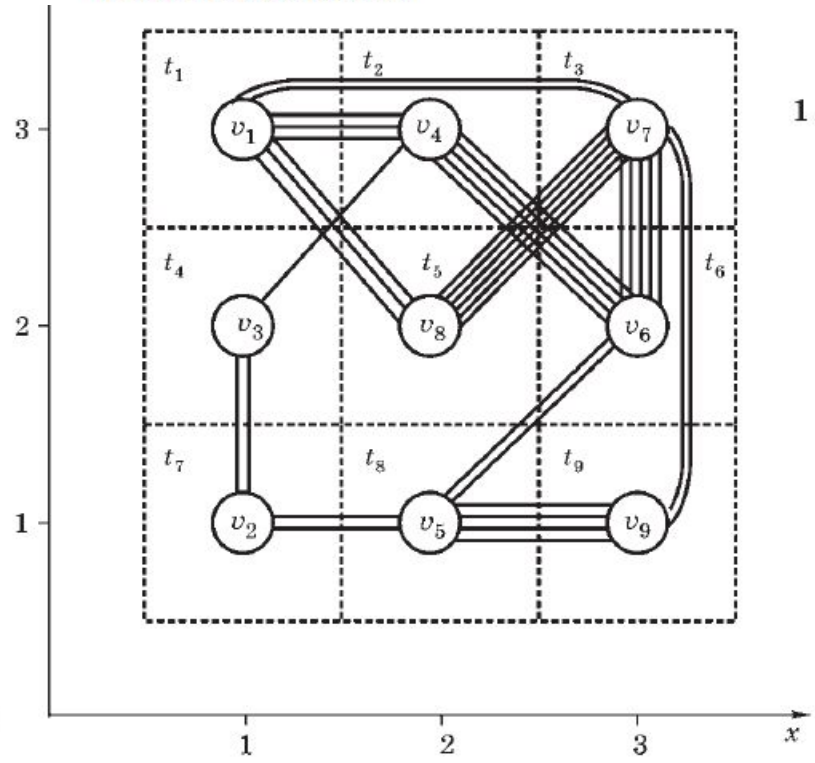



**Рис. 5.14**
Размещение элементов после первой итерации

# Задача размещения по слоям платы

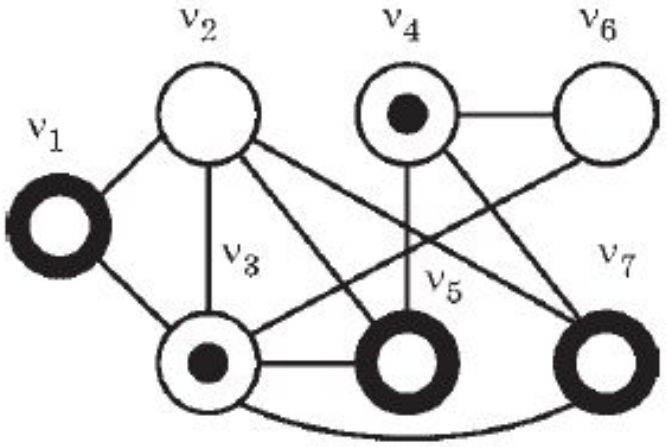



**Рис. 5.21**
Раскрашенный граф

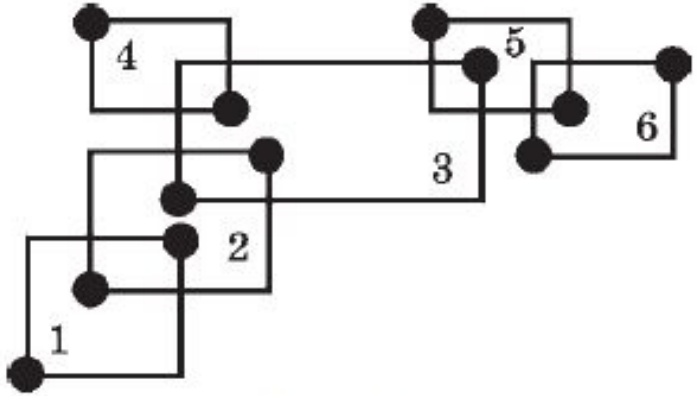



**Рис. 5.22**
Замена цепей схемы прямоугольниками

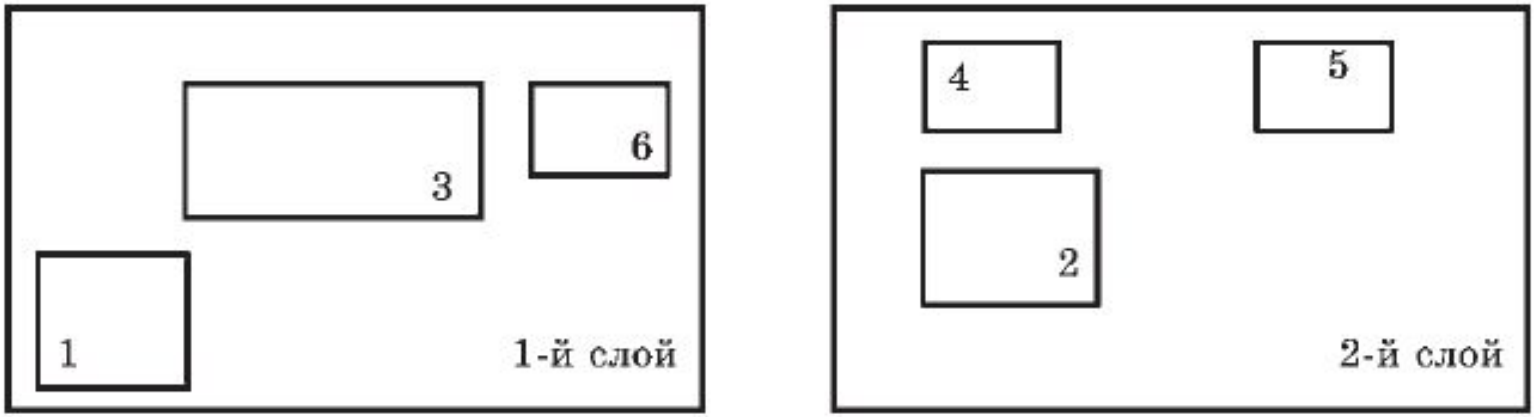



**Рис. 5.23**
Распределение цепей схемы по слоям

# Задача трассировки

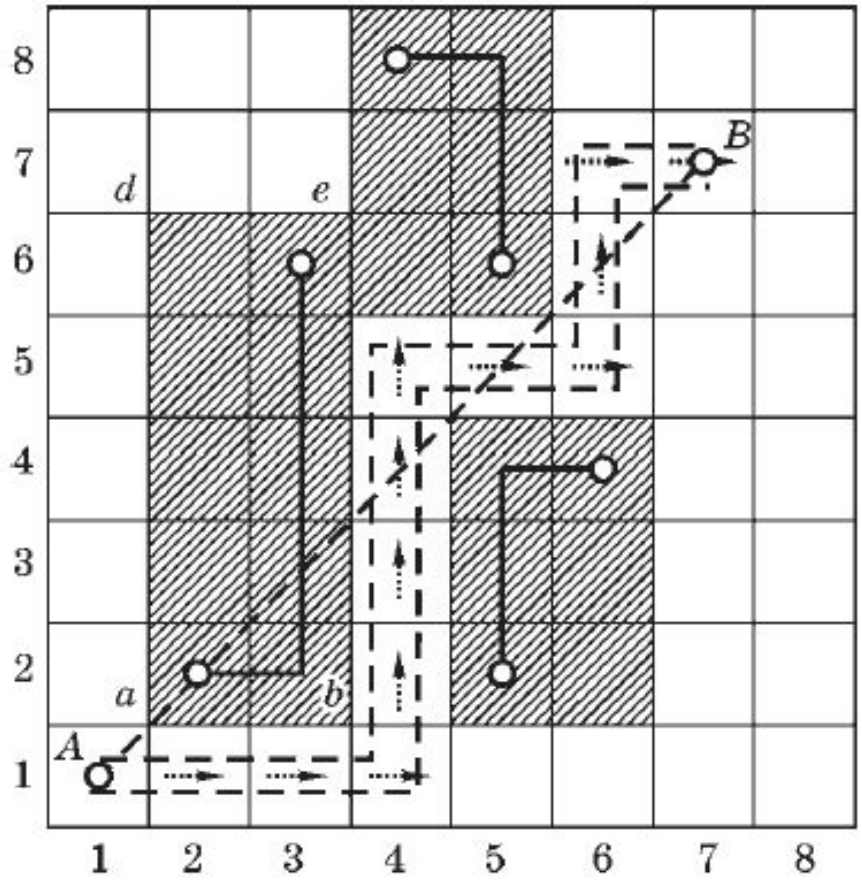



**Рис. 5.31**
Трассировка с минимальным отклонением от линии $AB$

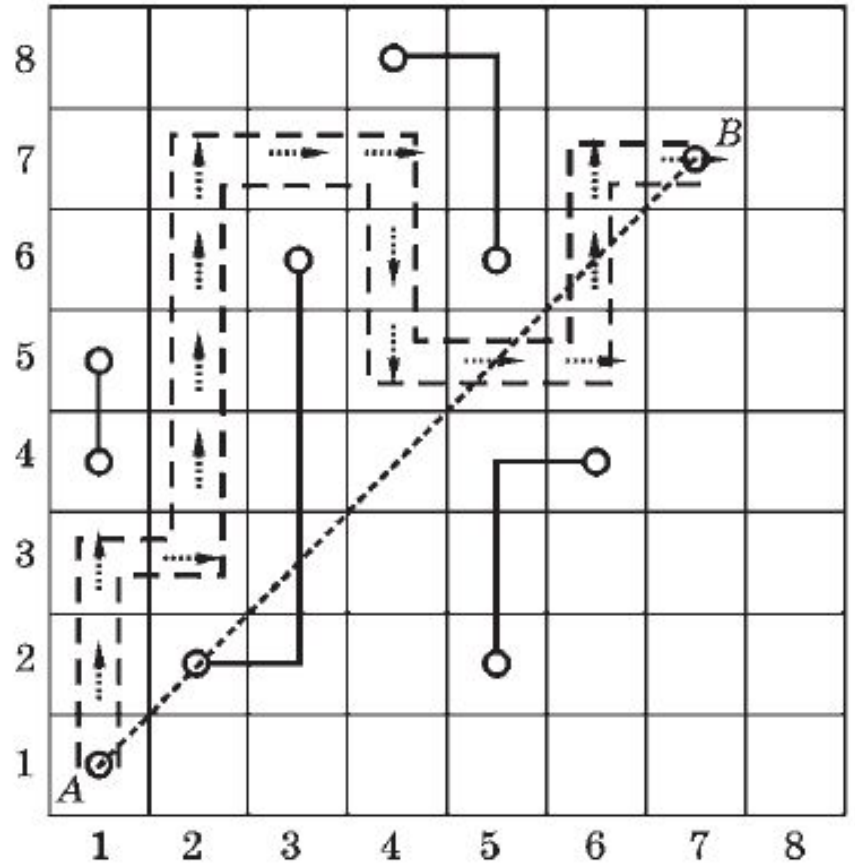



**Рис. 5.32**
Трассировка с обходом занятых дискрет

# Вопрос 3

## Методы поиска кратчайших маршрутов на графах

Известные алгоритмы **выбора оптимальных маршрутов** в зависимости от числа получаемых оптимальных маршрутов можно разделить на два класса. Первый из них составляют алгоритмы определения оптимальных маршрутов от заданной вершины до множества других вершин. В частном случае это множество может содержать все оставшиеся вершины. Алгоритмы, образующие второй класс, позволяют вычислять оптимальные маршруты между всеми парами вершин графа-модели сети. Такая классификация является условной в том смысле, что применение алгоритма, принадлежащего первому классу, ко всем вершинам эквивалентно выполнению алгоритма второго класса.

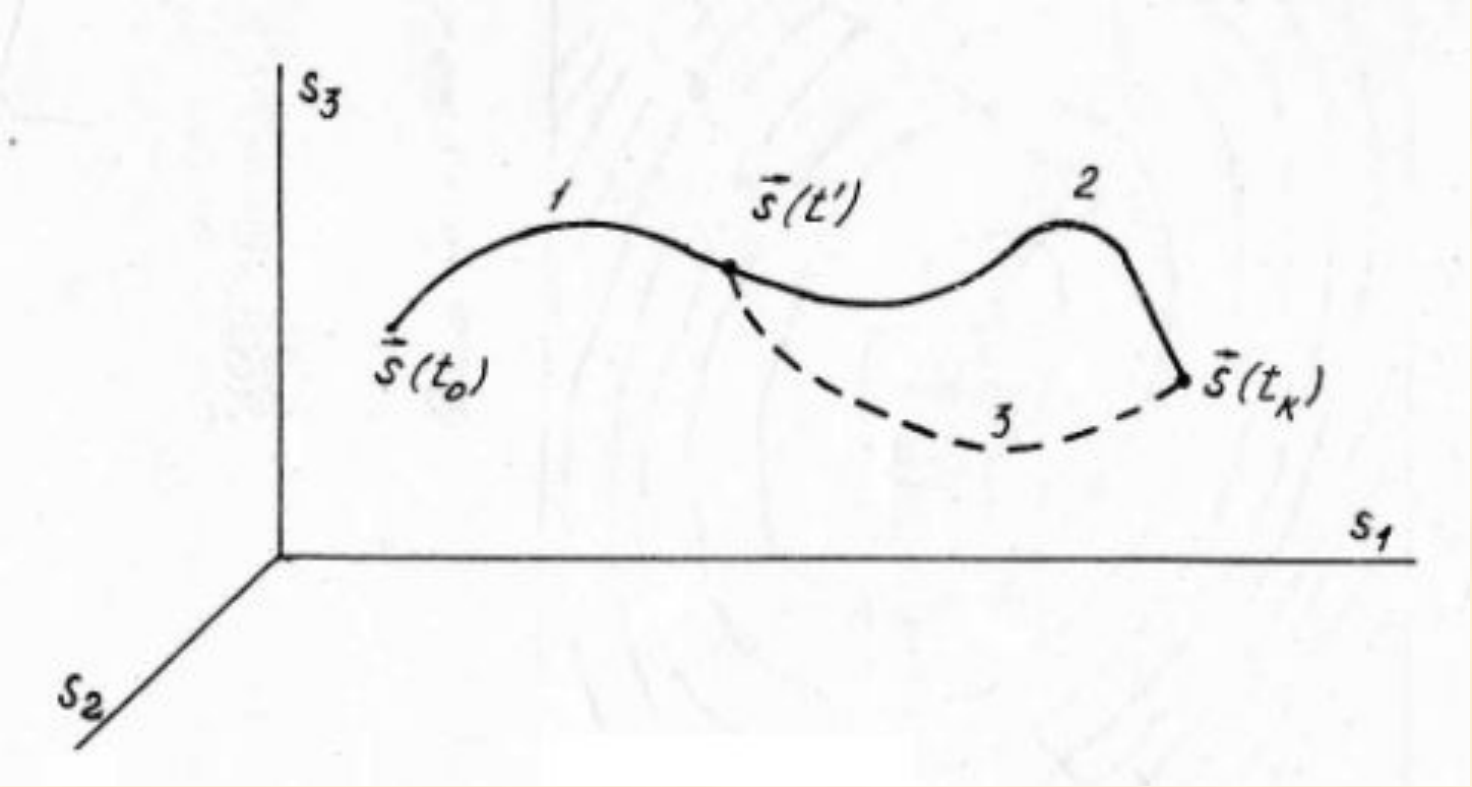
Наиболее известными представителями первого класса являются алгоритмы Дийкстры, Форда, двойного поиска, а второго – Флойда-Уоршола, Данцига, Уоррена, матричный и декомпозиционный алгоритмы. В основу построения этих алгоритмов положен принцип оптимальности динамического программирования, в соответствии с которым любой отрезок оптимального пути сам является оптимальным путем.

## ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Метод динамического программирования был предложен в 50-х годах Р. Беллманом для решения широкого класса оптимизационных задач. Он применим к системам, для которых справедлив принцип оптимальности (принцип Беллмана).

Система удовлетворяет принципу Беллмана, если она обладает марковскими свойствами: ее поведение на любом конечном отрезке времени $t_1 \leqslant t \leqslant t_k$ полностью определяется управлением на этом отрезке и состоянием системы в начальный момент времени $t_1$.

Принцип оптимальности часто применяется к таким системам, точное математическое описание которых неизвестно, а состояния и критерии оптимальности задаются набором экспериментальных значений в дискретные моменты времени (или на этапах, вообще не связанных со временем). В этих случаях справедливость принципа оптимальности доказать не удается и он, как правило, принимается интуитивно.

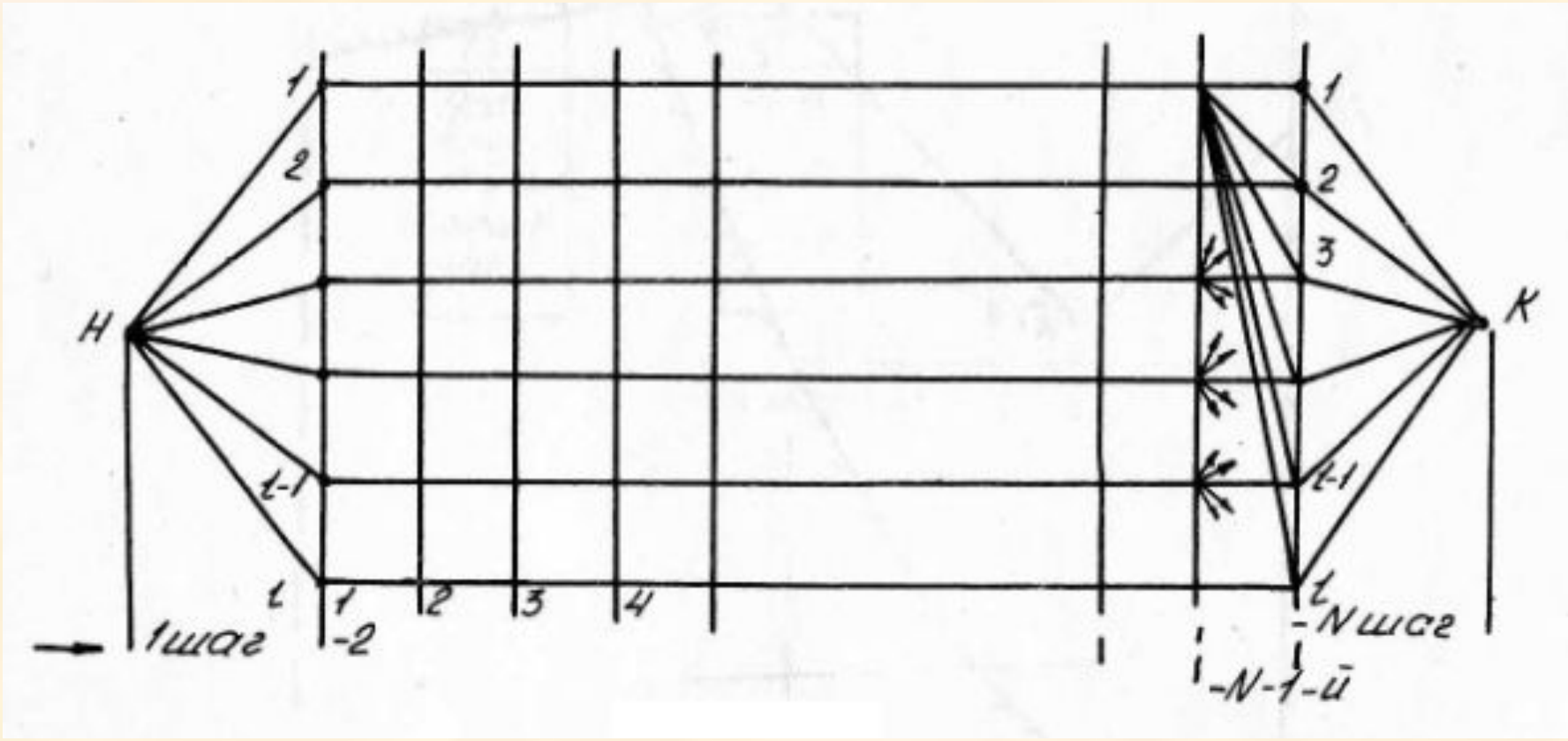

$s_3$
1
$\bar{s}(t')$
2
$\bar{s}(t_0)$
$\bar{s}(t_K)$
3
$s_1$
$s_2$

Решение задачи оптимального управления даже в простейшем случае довольно сложно. Оно в значительной степени упрощается при представлении ее в дискретной форме. Разобьем интервал времени $[t_0, t_к]$ на $N$ интервалов таких, что внутри каждого из них управление $x(t)$ можно считать постоянным. При этом задача управления на одном шаге предельно упрощается, поскольку становится скалярной. Естественно, что легче много раз решить скалярную задачу оптимизации, чем один раз — векторную. Кроме того, будем считать, что на каждом интервале имеет место всего $l$ вариантов различных состояний . Требуется перевести систему из начального состояния в конечное за $m$ шагов, оптимальным способом минимизируя

$$\Phi = \sum_{i=1}^{N} G_i(s_i,\ x_i).$$

На первый взгляд кажется, что задача может быть решена методом перебора. Просчитывая значения целевого функционала для каждого из возможных маршрутов, следует выбрать оптимальный из них. Однако легко видеть, что число маршрутов равно $l^N$ и при больших $l$ и $N$ решить задачу не представляется возможным.

Задачу решают как для непрерывного случая. Получают уравнение Беллмана с непрерывным временем, из которого однозначно следует алгоритм нахождения оптимальной траектории.



H
K
1
2
3
4
1шаг
-2
-N шаг
-N-1-й

В табл. 4.1. приведены данные по вычислительной сложности $(V)$, выраженной в виде зависимости числа элементарных операций от размерности сети $(N)$ ,.для наиболее известных алгоритмов.

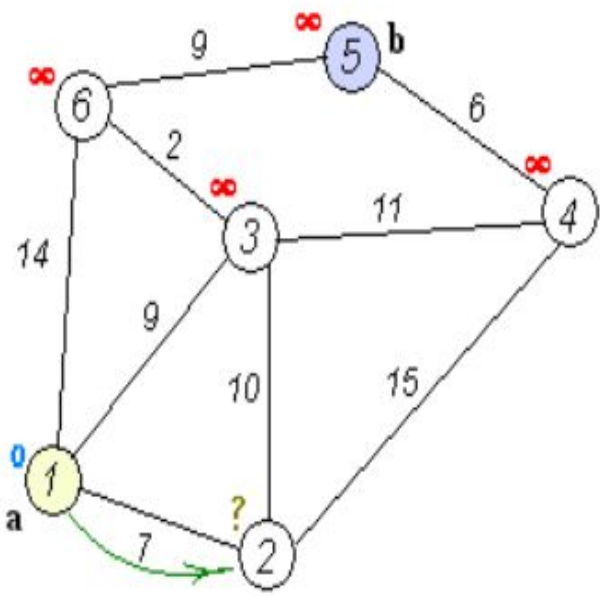
Таблица 4.1

| № | Алгоритмы | Вычислительная сложность |
|---|---|---|
| 1 | Дийкстры | $(N-1)(N-2)$ |
| 2 | Форда | $2(N-2)^2(N-1)$ |
| 3 | Двойного поиска | $2(N-2)^2(N-1)$ |
| 4 | Флойда-Уоршола | $2N(N^2-3N+4)$ |
| 5 | Уоррена | $2N^2(N-1)$ |
| 6 | Данцига | $N(N-1)(2N-1)$ |
| 7 | Матричный | $4N(N^2-3N+4)$ |

Из этой таблицы следует, что, несмотря на определенные отличия в виде функции $V = f(N)$, ее порядок остается неизменным для каждого класса алгоритмов и при достаточно больших значениях размерности сети вычислительную сложность можно оценивать для алгоритмов первого класса величиной $O(N^2)$, а для второго – $O(N^3)$. Следовательно, с помощью известных алгоритмов можно найти оптимальные маршруты между всеми парами вершин за время пропорциональное $O(N^3)$.

**Алгори́тм Де́йкстры** — алгоритм на графах, изобретенный Э. Дейкстрой. Находит кратчайшее расстояние от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании и сетевых технологиях, например, его использует протокол OSPF для устранения кольцевых маршрутов. Известен также под названием **Сначала Кратчайший Путь** (*Shortest Path First*).

Каждой вершине из *V* сопоставим метку — минимальное известное расстояние от этой вершины до *a*. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

**Инициализация**. Метка самой вершины *a* полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от *a* до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

**Шаг алгоритма**. Если все вершины посещены, алгоритм завершается. В противном случае из ещё не посещённых вершин выбирается вершина *u*, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых *u* является предпоследним пунктом. Вершины, в которые ведут рёбра из *u*, назовем соседями этой вершины. Для каждого соседа, кроме уже посещённых, рассмотрим новую длину пути, равную сумме текущей метки *u* и длины ребра, соединяющего *u* с этим соседом. Если полученная длина меньше метки соседа, заменим метку этой длиной. Рассмотрев всех соседей, пометим вершину *u* как посещенную и повторим шаг.